PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-322306
(43)Date of publication of application : 24.11.2000

(51)Int.Cl. G06F 12/00
G06F 13/00

(21)Application number : 11-143502
(22)Date of filing : 24.05.1999

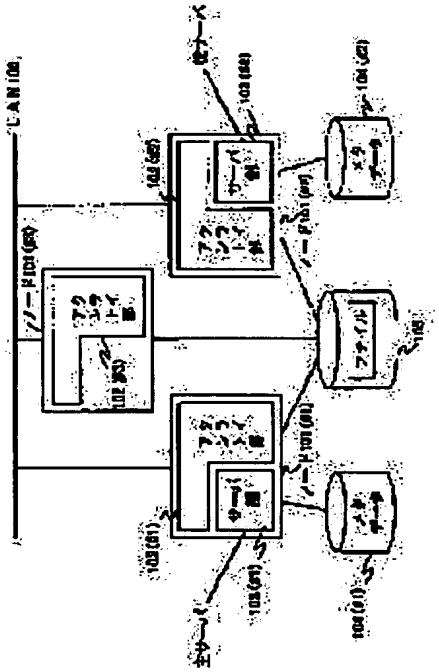
(71)Applicant : FUJITSU LTD
(72)Inventor : SHINKAI YOSHITAKE
TSUCHIYA YOSHIHIRO
MURAKAMI TAKEO

(30)Priority

Priority number	Priority date	Priority country
10328463	18.11.1998	JP
11063589	10.03.1999	JP

(54) INTER-NODE SHARED FILE CONTROL SYSTEM

(57)Abstract:
PROBLEM TO BE SOLVED: To reduce the increase of overhead and the deterioration of system performance accompanied by file access, and to prevent the inversion of a file time at the time of server switching in a duplex server system in an inter- node shared file system with tokens.
SOLUTION: At the time of requesting a token from a client part 102 to a server 103, the server part 103 responds with the token of a whole file to the client part 102 when any competition between the plural clients 102 is not generated. Only at the time of performing access to the final block of the file, the client part 102 captures the size token corresponding to the file from the server part 103, and performs access to the final block. The server part 103 can simultaneously respond with the time token of write authority to permit the change of the file time to the plural client parts 102. The client part 102 obtains the time token of the write authority, and then executes file access without inquiring for the file time to the server part 103. The server part 103 collects the time token of the write authority from the client part 102 in a prescribed timing, and updates the file time managed by itself.



LEGAL STATUS

[Date of request for examination] 29.10.2002

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]	
[Patent number]	3866448
[Date of registration]	13.10.2006
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－322306
（P2000－322306A）
(43)公開日　平成12年11月24日(2000.11.24)

(51)Int.Cl.⁷	識別記号	FI	テーマコード（参考）
G06F 12/00	535	G06F 12/00	535C 5B082
	531		531D 5B089
13/00	351	13/00	351E

審査請求　未請求　請求項の数35　ＯＬ　（全 27 頁）

(21)出願番号　特願平11－143502
(22)出願日　平成11年５月24日(1999.5.24)
(31)優先権主張番号　特願平10－328463
(32)優先日　平成10年11月18日(1998.11.18)
(33)優先権主張国　日本（ＪＰ）
(31)優先権主張番号　特願平11－63589
(32)優先日　平成11年３月10日(1999.3.10)
(33)優先権主張国　日本（ＪＰ）

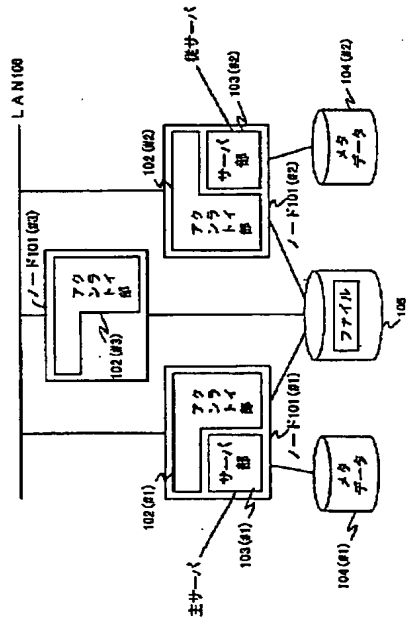
(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(72)発明者　新開　慶武
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　土屋　芳浩
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(74)代理人　100074099
弁理士　大菅　義之　（外１名）

最終頁に続く

(54)【発明の名称】　ノード間共用ファイル制御方式

(57)【要約】
【課題】　トークンによるノード間共用ファイルシステムにおいて、ファイルアクセスに伴うオーバヘッドの増大とシステム性能の低下を抑制し、二重化サーバシステムでのサーバ切替え時のファイル時刻の逆転を防止することにある。
【解決手段】　クライアント部１０２からサーバ部１０３へのトークンの要求時に、１０３は、複数の１０２間で競合が無ければ、１０３から１０２へファイル全体のトークンを応答する。１０２は、ファイルの最終ブロックへのアクセス時のみ、１０３からそのファイルに対応するサイズトークンを獲得した上でその最終ブロックにアクセスする。１０３は、ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の１０２に同時に応答できる。１０２は、write 権の時刻トークンを獲得した後は、１０３にファイル時刻を問い合わせることなく、ファイルアクセスを実行する。１０３は、所定のタイミングで、１０２からwrite 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する。

本発明の実施の形態のシステム構成図



【特許請求の範囲】
【請求項１】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記クライアント装置から前記サーバ装置へのトークンの要求時に、前記サーバ装置において複数の前記クライアント装置間での該トークンの競合の有無を判定し、
該競合が無ければ、前記サーバ装置から前記クライアント装置へファイル全体のトークンを応答する、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項２】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、
前記クライアント装置から前記サーバ装置へのトークンの要求時に、複数の前記クライアント装置間での該トークンの競合の有無を判定する判定手段と、
該判定手段が該競合が無いと判定した場合に、前記クライアント装置へファイル全体のトークンを応答する応答手段と、
を含むことを特徴とするサーバ装置。
【請求項３】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記クライアント装置から前記サーバ装置へのトークンの要求時に、複数の前記クライアント装置間での該トークンの競合の有無を判定する機能と、
該判定手段が該競合が無いと判定した場合に、前記クライアント装置へファイル全体のトークンを応答する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項４】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記クライアント装置と前記サーバ装置の間で、ファイル時刻を制御するための時刻トークンを通信し、
前記サーバ装置において、前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行し、
前記クライアント装置において、前記write 権の時刻トークンを獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行し、
前記サーバ装置において、所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項５】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、
前記サーバ装置との間で、ファイル時刻を制御するための時刻トークンを通信する通信手段と、
前記サーバ装置から複数の前記クライアント装置に同時に応答され得るトークンであって前記ファイル時刻の変更を許容するwrite 権の時刻トークンを前記サーバ装置から獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行するアクセス制御手段と、
を含むことを特徴とするクライアント装置。
【請求項６】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記サーバ装置との間で、ファイル時刻を制御するための時刻トークンを通信する機能と、
前記サーバ装置から複数の前記クライアント装置に同時に応答され得るトークンであって前記ファイル時刻の変更を許容するwrite 権の時刻トークンを前記サーバ装置から獲得した後は、前記サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスを実行する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項７】　ユーザプログラムからのファイル操作要

求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、
前記クライアント装置との間で、ファイル時刻を制御するための時刻トークンを通信する通信手段と、
前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行する応答手段と、
所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新するファイル時刻更新手段と、
を含むことを特徴とするサーバ装置。
【請求項８】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記クライアント装置との間で、ファイル時刻を制御するための時刻トークンを通信する機能と、
前記ファイル時刻の変更を許容するwrite 権の時刻トークンを複数の前記クライアント装置に同時に応答する制御を実行する機能と、
所定のタイミングで前記クライアント装置から前記write 権の時刻トークンを回収し、自身が管理するファイル時刻を更新する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項９】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記クライアント装置と前記サーバ装置の間で、ファイルサイズの拡張を制御するためのサイズトークンを通信し、
前記クライアント装置において、ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスする、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項１０】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、
前記サーバ装置との間で、ファイルサイズの拡張を制御するためのサイズトークンを通信する通信手段と、
ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスするアクセス手段と、
を含むことを特徴とするクライアント装置。
【請求項１１】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記サーバ装置との間で、ファイルサイズの拡張を制御するためのサイズトークンを通信する機能と、
ファイルの最終ブロックにアクセスする場合においてのみ、前記サーバ装置から該ファイルに対応するサイズトークンを獲得した上で該最終ブロックにアクセスする機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項１２】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記クライアント装置と前記サーバ装置の間で、ファイルデータのアクセスを制御するためのデータトークンを通信し、
該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項１３】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライ

アント装置であって、
前記サーバ装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する第１の通信手段と、
該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する第２の通信手段と、
を含むことを特徴とするクライアント装置。
【請求項１４】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記サーバ装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する機能と、
該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項１５】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であって、
前記クライアント装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する第１の通信手段と、
該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する第２の通信手段と、
を含むことを特徴とするサーバ装置。
【請求項１６】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該サーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記クライアント装置との間で、ファイルデータのアクセスを制御するためのデータトークンを通信する機能と、
該データトークンの通信時に、該データトークンに対応するファイルのディスク上での位置を示すエクステント情報を通信する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項１７】　請求項１、２、４、５、７、９、１０、１２、１３、又は１５のいずれか１項に記載のノード間共用ファイルシステムにおいて前記サーバ装置が二重化される構成を有するノード間共用ファイル制御方法であって、
主系のサーバ装置においてファイル時刻が設定される際に、該ファイル時刻を従系のサーバ装置に送信し、
該従系のサーバ装置において、該ファイル時刻を設定する、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項１８】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記サーバ装置において、複数のノードから共用される１つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理し、
前記クライアント装置において、前記サーバ装置に対して、ディスク領域のリザーブを要求し、
該リザーブ要求に対して、前記サーバ装置において、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理し、
前記リザーブ要求を発行したクライアント装置において、該リザーブ要求に応答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理し、
前記クライアント装置において、ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、該クライアント装置が管理する前記リザーブ中ディスク領域群から最適なリザーブ中ディスク領域を選択し、そこに対してデータ書出しを実行し、該リザーブ中ディスク領域を前記リザーブ中ディスク領域群としての管理からはずし、該データ書出しを実行したリザーブ中ディスク領域に関する情報を前記サーバ装置に通知し、
前記サーバ装置において、前記クライアント装置から通知された情報に対応する前記データ書出しが発生したリ

ザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理する、
過程を含むことを特徴とするノード間共用ファイル制御方法。
【請求項１９】　請求項１８に記載の方法であって、
前記サーバ装置における前記空きディスク領域群及びリザーブ中ディスク領域群の管理と、前記クライアント装置における前記リザーブ中ディスク領域群の管理を、前記ディスク領域の複数のサイズ範囲毎に行う、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２０】　請求項１８に記載の方法であって、
前記クライアント装置において、前記ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、該クライアント装置が管理する前記リザーブ中ディスク領域群から前記ファイルへのデータ書出しが既に行われているディスク領域に連続するリザーブ中ディスク領域を選択し、該選択に失敗した場合には、前記サーバ装置に対して、該連続するリザーブ中ディスク領域のリザーブ要求を発行する、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２１】　請求項１８に記載の方法であって、
前記サーバ装置において、前記クライアント装置の障害を監視し、その結果障害が検出されたクライアント装置に対応する前記リザーブ中ディスク領域群を、全て前記空きディスク領域群に変更する、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２２】　請求項１８に記載の方法であって、
前記クライアント装置において、それが管理する前記リザーブ中ディスク領域群中のディスク領域が所定量を下回ったときに、前記サーバ装置に対して、新たなディスク領域のリザーブ要求を発行する、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２３】　請求項１８に記載の方法であって、
前記クライアント装置において、前記ユーザプログラムによるファイルへのデータ書出し要求に基づいて書き出されるデータを、主記憶上にキャッシュし、前記リザーブ中ディスク領域の割り当てを遅延させる、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２４】　請求項１８に記載の方法であって、
前記クライアント装置において、前記データ書出しを実行したリザーブ中ディスク領域に関する情報の前記サーバ装置への通知は、前記ユーザプログラムがファイルをクローズし、又はキャッシュが一杯になり、或いは前記サーバ装置からデータトークンの回収を要求されるタイミングで行う、
過程を更に含むことを特徴とするノード間共用ファイル制御方法。
【請求項２５】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であって、
前記サーバ装置に対して、ディスク領域のリザーブを要求するリザーブ要求手段と、
該リザーブ要求に応答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理するリザーブ中ディスク領域群管理手段と、
ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、前記リザーブ中ディスク領域群管理手段が管理する前記リザーブ中ディスク領域群から最適なリザーブ中ディスク領域を選択し、そこに対してデータ書出しを実行し、前記リザーブ中ディスク領域群管理手段に対して該データ書出しを実行したリザーブ中ディスク領域の管理をはずさせ、該リザーブ中ディスク領域に関する情報を前記サーバ装置に通知するクライアント側データ書出し制御手段と、
を含むことを特徴とするクライアント装置。
【請求項２６】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成する当該クライアント装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
前記サーバ装置に対して、ディスク領域のリザーブを要求する機能と、
該リザーブ要求に応答して前記サーバ装置から通知された情報に対応するリザーブ中ディスク領域をリザーブ中ディスク領域群として管理する機能と、
ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、自身が管理する前記リザーブ中ディスク領域群から最適なリザーブ中ディスク領域を選択し、そこに対してデータ書出しを実行し、該リザーブ中ディスク領域を前記リザーブ中ディスク領域群としての管理からはずし、該データ書出しを実行したリザーブ中ディスク領

域に関する情報を前記サーバ装置に通知する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項２７】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成するサーバ装置であって、
複数のノードから共用される１つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理するディスク領域管理手段と、
前記クライアント装置からのディスク領域のリザーブ要求に対して、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理するリザーブ中ディスク領域確保手段と、
前記クライアント装置から通知された情報に対応するデータ書出しが発生したリザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理するサーバ側データ書出し制御手段と、
を含むことを特徴とするサーバ装置。
【請求項２８】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイルシステムを構成するサーバ装置であるコンピュータにより使用されたときにそれによって読み出されるプログラムを記録した記録媒体であって、
複数のノードから共用される１つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び前記各クライアント装置に対応するリザーブ中ディスク領域群を管理する機能と、
前記クライアント装置からのディスク領域のリザーブ要求に対して、前記空きディスク領域群からディスク領域をリザーブ中ディスク領域として確保し、それに関する情報を該リザーブ要求を発行したクライアント装置に通知すると共に、該確保したリザーブ中ディスク領域を前記リザーブ要求を発行したクライアント装置に対応する前記リザーブ中ディスク領域群として管理する機能と、
前記クライアント装置から通知された情報に対応するデータ書出しが発生したリザーブ中ディスク領域を、該通知を行ったクライアント装置に対応する前記リザーブ中ディスク領域群としての管理からはずして前記使用中ディスク領域として管理する機能と、
を前記コンピュータに行わせるためのプログラムを記録したコンピュータ読出し可能記録媒体。
【請求項２９】　ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上で該ファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御方法であって、
前記サーバ装置において、前記クライアント装置からのトークン回収完了メッセージの受信時に、該メッセージに対応するトークン回収の契機となった要求を処理している実行単位が保持していたファイルロックを継承して処理を実行することによりデッドロックを回避する過程を含む、
ことを特徴とするノード間共用ファイル制御方法。
【請求項３０】　請求項２９に記載の方法であって、
前記ロックの継承を行える実行単位を１つに制限する過程を更に含む、
ことを特徴とするノード間共用ファイル制御方法。
【請求項３１】　請求項２９に記載の方法であって、
前記トークン回収の待ち状態を資源として記憶し、他の資源の獲得待ち状態との関係から、デッドロック状態を自動的に検出する過程を更に含む、
ことを特徴とするノード間共用ファイル制御方法。
【請求項３２】請求項３１に記載の方法であって、
前記デッドロック状態が検出され該状態の原因となっているトランザクションがキャンセルさせられる際に、更新されたキャッシュデータの無効化と共に、主記憶装置に常駐されている関連制御表の再設定を行う過程を更に含む、
ことを特徴とするノード間共用ファイル制御方法。
【請求項３３】　請求項２９に記載の方法であって、
デッドロック状態の発生に備え、ファイル又はディスクに関する属性情報を保持するメタデータの更新をキャッシュ上でのみ行い、ディスクへの書き込みが、要求された処理の完了まで遅延させられるトランザクション制御において、
キャッシュデータの更新時に更新されたキャッシュ位置を記録する過程と、
トランザクションの完了時に、前記記録から必要最小限の変更データのみをログファイルに書き出すことによりログデータ量を削減する過程と、
を含むことを特徴とするノード間共用ファイル制御方法。
【請求項３４】　請求項３３に記載の方法であって、
更新されたキャッシュ位置を記録する際に、該記録と先

行する記録とをマージすることにより、ログファイルに書き出すログデータ量を最小化する過程を更に含む、
ことを特徴とするノード間共用ファイル制御方法。
【請求項３５】　請求項３３に記載の方法であって、
前記キャッシュは２次キャッシュを含む、
ことを特徴とするノード間共用ファイル制御方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、複数のノード（ホストコンピュータ）から同一のファイルを共用することを可能とするノード間共用ファイルシステム（分散ファイルシステム）のコンシステンシ保証制御技術に関する。
【０００２】
【従来の技術】分散ファイルシステムにおいて、トークンを利用して複数のノード上にキャッシュされているデータのコンシステンシ（一貫性、整合性）を保つ方式は良く知られている。代表的な方式では、ファイルのアクセス範囲（通常、ブロック番号の始端と終端が用いられる）ごとにmultiple-read/single-writeの制御を行うトークンが用意される。そして、ファイルにアクセスしようとするノードは、自身がアクセス範囲のトークンを保持しているか否かを調べ、もし保持していなければトークンを管理しているサーバにトークンを要求する。トークンを管理しているサーバは、read権は複数のノードに渡されることを許し（multiple-read ）、write 権は１つのノードのみに渡されるように（single-write）、アクセス権制御を実行する。
【０００３】
【発明が解決しようとする課題】上述の従来方式は、各ノードにキャッシュされているデータの一貫性を保ちつつサーバとクライアントの間の通信を減らすために有効な方式であるが、以下の問題点を有する。
１）ファイルアクセスの都度にトークンを獲得する必要がある。例えば、科学技術計算のための巨大なファイルをユーザがシーケンシャルにアクセスする場合、ユーザは、特定バイトずつのファイルアクセス要求を出す都度に、サーバにトークンを獲得するための要求を発行せざるを得ない。この事実は、オーバヘッドの増大を招く。
２）ファイルが最後にアクセスされた時刻を保持するファイルアクセス時刻（ファイル時刻）の正当性を保証するために、ユーザはファイルアクセス要求を発行する都度にサーバにそのアクセスの存在を通知せざるを得ない。この事実は、オーバヘッドの増大を招く。
３）ユーザはファイルサイズを更新するときにはその旨をサーバに通知し、サーバは他ノードに発行されている全てのトークンを回収しなければならない。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることができず、システム全体の性能が低下するといった問題が生ずる。
４）サーバが二重化され、障害発生時に運用サーバが待機系サーバに切り替えられる機能を有するシステムにおいて、待機系サーバへの切替えの時点でいままで運用されてきた時計も待機系のサーバ内の時計に切り替えられるため、ファイル時刻の逆転現象が発生する可能性がある。この事実は、データのコンシステンシの喪失を招く。
５）メインフレームで採用されるような、ディスクがノード間で直接共用されネットワークを介したデータ転送が削減される方式を、離散ファイルを特徴とするオープン系のファイルシステムに適用しようとした場合に、各ノードはファイルシステム上でブロックを割り当てる都度にサーバと通信する必要が生ずる。この事実は、オーバヘッドの増大を招く。
一方、トークンを利用した分散ファイルシステムにおいては、複数のノードが同時並行的なアクセスを行うため、ファイルシステムの耐故障性に関しても十分な配慮が必要である。一般に、ファイルシステムの耐故障性を向上させる方式として、ログファイルを設けてメタデータの更新をトランザクショナルに行うログ方式が知られている。ログ方式では一般に、１つのトランザクションの処理途中結果を他のトランザクションに見せてはならないという制約のために、いわゆる２フェーズロック制御が行われる。この制御では、更新に必要なロックが順に獲得されてゆき、全ての更新が完了した時点で一括して、メタデータの更新内容がログファイルロックに書き出され、書出しが完了した時点でロックが一括して返却される。この際に必然的に発生する複数のロック獲得に伴うデッドロックは、資源獲得を示す有向グラフを用いて自動的に検出され、デッドロックの原因となっている一方のトランザクションがキャンセルされ、再試行させられることにより解消される方式が、一般的に用いられる。
【０００４】しかし、上述のようなログ方式をトークンシステムに適用してデッドロックを自動的に検出し回復を図る汎用的な方式は考え出されていない。また、従来のログ方式では、ログがキャッシュブロック単位に採取されると共に、トランザクション終了時にファイルシステムの実更新が発生するため、Ｉ／Ｏ量が相対的に多くなるという欠陥があった。
【０００５】また上記ログ方式では、トランザクションのキャンセル時のデータ復元処理がメタデータのみに限られ、性能向上のために用意されたメモリに常駐する制御表は対象外であるため、プログラム作成が難しいという欠陥も持っていた。
【０００６】本発明の課題は、トークンを用いたノード間共用ファイルシステムにおいて、上述の各問題点を解決することにあり、ファイルアクセスに伴うオーバヘッドの増大とシステム性能の低下を抑制すると共に、二重

化サーバシステムにおけるサーバ切替え時のファイル時刻の逆転を防止し、更にメタデータの更新をコンシステントにかつデッドロックフリーで行なうことにより従来のログ方式の性能上及びプログラム作成上の問題点を解決することにある。

【０００７】

【課題を解決するための手段】本発明は、ユーザプログラムからのファイル操作要求を受けて、１つのノード内のクライアント装置がそれと同一の又は他のノード内のサーバ装置からトークンを獲得した上でそのファイル操作要求を処理することにより、複数のノードからの同一ファイルの共用を可能とするノード間共用ファイル制御システムを前提とする。

【０００８】本発明の第１の態様は、以下の構成を有する。まず、クライアント装置からサーバ装置へのトークンの要求時に、サーバ装置において複数のクライアント装置間でのそのトークンの競合の有無が判定される。

【０００９】そして、その競合が無ければ、サーバ装置からクライアント装置へファイル全体のトークンが応答される。以上の構成を有する本発明の第１の態様の構成では、例えばopen要求時等においてファイル全体のトークンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、１つのノードからのwrite 要求の発行時に他のノードからread命令が発行される確率は小さい。従って、１つのノードに引き渡されたファイル全体のトークンが回収される確率も低く、ファイルへの連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

【００１０】本発明の第２の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイル時刻を制御するための時刻トークンが通信される。

【００１１】また、サーバ装置において、ファイル時刻の変更を許容するwrite 権の時刻トークンを複数のクライアント装置に同時に応答する制御が実行される。また、クライアント装置において、write 権の時刻トークンが獲得された後は、サーバ装置にファイル時刻を問い合わせることなく、ファイルアクセスが実行される。

【００１２】また、サーバ装置において、所定のタイミングでクライアント装置からwrite権の時刻トークンが回収され、自身が管理するファイル時刻が更新される。以上の構成を有する本発明の第２の態様の構成では、クライアント装置は、ユーザプログラムが１つのファイルに連続アクセスするような場合において、そのファイルへの最終的なアクセスが終了するまでwrite 権の時刻トークンを返却する必要もまたアクセスの有無をサーバ部に通知する必要もなく、他のノードとの間でそのファイルのファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能となる。

【００１３】本発明の第３の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイルサイズの拡張を制御するためのサイズトークンが通信される。

【００１４】そして、クライアント装置において、ファイルの最終ブロックにアクセスする場合においてのみ、サーバ装置からそのファイルに対応するサイズトークンが獲得された上でその最終ブロックにアクセスされる。

【００１５】以上の構成を有する本発明の第３の態様の構成では、ファイルの最終ブロックにアクセスするのでなければ、サイズトークンを獲得することなくファイルにアクセスすることが可能となり、これと並行して、他のノードは、サイズトークンを獲得してファイルの最終ブロックにアクセスし、ファイルのサイズを拡張するwrite 操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることが可能となり、システム全体の性能を向上させることができる。

【００１６】本発明の第４の態様は、以下の構成を有する。まず、クライアント装置とサーバ装置の間で、ファイルデータのアクセスを制御するためのデータトークンが通信される。

【００１７】そして、そのデータトークンの通信時に、そのデータトークンに対応するファイルのディスク上での位置を示すエクステント情報が通信される。以上の構成を有する本発明の第４の態様の構成では、複数のノードは、ディスク装置内のファイルに、ＬＡＮ経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

【００１８】本発明の第５の態様は、上述した本発明の第１乃至第４のいずれかの態様の構成を前提として、さらにサーバ装置が二重化される構成を前提とし、以下の構成を有する。

【００１９】まず、主系のサーバ装置においてファイル時刻が設定される際に、そのファイル時刻が従系のサーバ装置に送信される。そして、その従系のサーバ装置において、そのファイル時刻が設定される。

【００２０】以上の構成を有する本発明の第５の態様の構成では、サーバ切替時にも、矛盾のないファイル時刻の付与が可能となる。本発明の第６の態様は、以下の構成を有する。

【００２１】まず、サーバ装置において、複数のノードから共用される１つ以上のディスクボリューム毎に、空きディスク領域群、使用中ディスク領域群、及び各クライアント装置に対応するリザーブ中ディスク領域群が管理される。このとき、空きディスク領域群の管理が、ディスク領域の複数のサイズ範囲毎に行われるように構成することができる。

【００２２】次に、クライアント装置において、サーバ装置に対して、ディスク領域のリザーブが要求される。このとき、クライアント装置において、それが管理するリザーブ中ディスク領域群中のディスク領域が所定量を下回ったときに、サーバ装置に対して、新たなディスク領域のリザーブ要求が発行されるように構成することができる。

【００２３】次に、そのリザーブ要求に対して、サーバ装置において、空きディスク領域群からディスク領域がリザーブ中ディスク領域として確保され、それに関する情報がそのリザーブ要求を発行したクライアント装置に通知されると共に、その確保されたリザーブ中ディスク領域がリザーブ要求を発行したクライアント装置に対応するリザーブ中ディスク領域群として管理される。

【００２４】続いて、リザーブ要求を発行したクライアント装置において、そのリザーブ要求に応答してサーバ装置から通知された情報に対応するリザーブ中ディスク領域がリザーブ中ディスク領域群として管理される。このとき、クライアント装置において、リザーブ中ディスク領域群の管理が、ディスク領域の複数のサイズ範囲毎に行われるように構成することができる。

【００２５】更に、クライアント装置において、ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、そのクライアント装置が管理するリザーブ中ディスク領域群から最適なリザーブ中ディスク領域が選択され、そこに対してデータ書出しが実行され、そのリザーブ中ディスク領域がリザーブ中ディスク領域群としての管理からはずされ、そのデータ書出しを実行したリザーブ中ディスク領域に関する情報がサーバ装置に通知される。この通知は、ユーザプログラムがファイルをクローズし、又はキャッシュが一杯になり、或いはサーバ装置からデータトークンの回収を要求されるタイミングで行うように構成することができる。このとき、ユーザプログラムによるファイルへのデータ書出し要求に基づいて書き出されるデータが、主記憶上にキャッシュされ、リザーブ中ディスク領域の割り当てが遅延させられるように構成することができる。

【００２６】そして、サーバ装置において、クライアント装置から通知された情報に対応するデータ書出しが発生したリザーブ中ディスク領域が、その通知を行ったクライアント装置に対応するリザーブ中ディスク領域群としての管理からはずされて使用中ディスク領域として管理される。

【００２７】上述の本発明の第６の態様の構成において、クライアント装置において、ユーザプログラムによるファイルへのデータ書出し要求に伴って新たなディスク領域を割り当てる必要が発生した場合に、そのクライアント装置が管理するリザーブ中ディスク領域群からファイルへのデータ書出しが既に行われているディスク領域に連続するリザーブ中ディスク領域が選択され、その選択に失敗した場合には、サーバ装置に対して、その連続するリザーブ中ディスク領域のリザーブ要求が発行されるように構成することができる。

【００２８】また、サーバ装置において、クライアント装置の障害が監視され、その結果障害が検出されたクライアント装置に対応するリザーブ中ディスク領域群が、全て空きディスク領域群に変更されるように構成することができる。

【００２９】以上の構成を有する本発明の第６の態様の構成では、クライアント装置は、サーバ装置に問い合わせることなく、新たなディスク領域をファイルに割り当てることが可能となる。このため、クライアント装置とサーバ装置との間の通信回数を削減でき、システム全体の性能を向上させることが可能となる。また、新たに割り当てられたディスク領域は、データが書き込まれた後のクライアント装置からサーバ装置への応答によって初めて、そのファイルのメタデータ等として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

【００３０】本発明の第７の構成は、サーバ装置において、クライアント装置からのトークン回収完了メッセージの受信時に、そのメッセージに対応するトークン回収の契機となった要求を処理している実行単位が保持していたファイルロックを継承して処理を実行することによりデッドロックを回避する過程を含むように構成される。この場合に、ロックの継承を行える実行単位を１つに制限する過程を更に含むように構成することができる。

【００３１】上述した本発明の第７の構成によれば、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。本発明の第８の構成は、本発明の第７の構成において、トークン回収の待ち状態を資源として記憶し、他の資源の獲得待ち状態との関係から、デッドロック状態を自動的に検出する過程を更に含むように構成される。

【００３２】上述した本発明の第８の構成によれば、トークンに基づいてトランザクション制御されているメタデータ等の更新処理におけるデッドロックの発生を適切に検出することができる。

【００３３】本発明の第９の構成は、本発明の第８の構成において、デッドロック状態が検出されその状態の原因となっているトランザクションがキャンセルさせられる際に、更新されたキャッシュデータの無効化と共に、主記憶装置に常駐されている関連制御表の再設定を行う過程を更に含むように構成される。

【００３４】上述した本発明の第９の構成によれば、トランザクションのキャンセルに伴う常駐制御表の高速なリストアが実現される。本発明の第１０の構成は、本発明の第７の構成を前提として、デッドロック状態の発生

に備え、ファイル又はディスクに関する属性情報を保持するメタデータの更新をキャッシュ上でのみ行い、ディスクへの書き込みが、要求された処理の完了まで遅延させられるトランザクション制御において、キャッシュデータの更新時に更新されたキャッシュ位置を記録する過程と、トランザクションの完了時に、前記記録から必要最小限の変更データのみをログファイルに書き出すことによりログデータ量を削減する過程とを含むように構成される。ここで、更新されたキャッシュ位置を記録する際に、その記録と先行する記録とをマージすることにより、ログファイルに書き出すログデータ量を最小化する過程を更に含むように構成することができる。

【００３５】上述した本発明の第１０の構成によれば、ログファイルに書き出されるログデータ量の削減が実現される。本発明の第１１の構成は、本発明の第１０の構成において、キャッシュが２次キャッシュを含むように構成される。

【００３６】上述した本発明の第１１の構成によれば、ログファイルを実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。

【００３７】

【発明の実施の形態】以下、本発明の実施の形態について詳細に説明する。図１は、本発明の実施の形態の構成を示すブロック構成図である。

【００３８】＃１～＃３の各ノード１０１は、ファイル１０５が格納されているディスク装置と直結され、またローカルエリアネットワーク（ＬＡＮ）１０６によって相互に接続される。

【００３９】ファイル１０５を共用する複数のノード１０１（図中では、＃１～＃３）の全てにクライアント部１０２、そのうちの２つのノード１０１（図中では、＃１と＃２）にサーバ部１０３が存在する。

【００４０】一方のノード１０１（＃１）内のサーバ部１０３（＃１）は主サーバ、他方のノード１０１（＃２）のサーバ部１０３（＃２）は従サーバと呼ばれる。それぞれのノード１０１内のクライアント部１０２は、主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）とのみ通信することにより、ファイル操作処理を実行する。

【００４１】主サーバであるサーバ部１０３（＃１）は、任意のクライアント部１０２からの要求（依頼）を処理して、その処理結果を、自身が保持するメタデータ１０４（＃１）に反映させる。従サーバであるノード１０１（＃２）内のサーバ部１０３（＃２）が存在するときには、主サーバであるサーバ部１０３（＃１）は、メタデータ１０４（＃１）の更新内容（差分）をサーバ部１０３（＃２）にも送る。従サーバであるサーバ部１０３（＃２）は、送られてきたデータをノード１０１（＃２）内のメタデータ１０４（＃２）に反映させる。

【００４２】任意のノード１０１内のクライアント部１０２は、図２に示されるように、そのノード１０１内のオペレーティングシステム（ＯＳ）２０１内に存在し、そのノード１０１内のユーザプログラム２０２からのファイル操作要求を、主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）の助けを借りて処理する。＃１又は＃２のノード１０１内のサーバ部１０３は、そのノード１０１内のオペレーティングシステム２０１に組み込んでもよいし、ユーザデーモンプログラムとしてオペレーティングシステム２０１の外に実装してもよい。このサーバ部１０３は、複数のノード１０１上のクライアント部１０２からのファイル操作要求を、ＬＡＮ１０６（図１参照）を介して受け付ける。

【００４３】上述の構成のもとでクライアント部１０２とサーバ部１０３がファイル操作制御を実行する場合、本実施の形態では、下記のトークンが用いられる。

１）ファイル１０５ごとに複数種類（例えば４種類）のトークンが用意され、その中に、ファイルサイズの拡張を制御しmultiple-read/single-write特性を有するサイズトークンが含めさせられる。

２）ファイル１０５ごとに複数種類（例えば４種類）のトークンが用意され、その中に、ファイル時刻を制御しmultiple-write/multiple-read特性を有する時刻トークンが含めさせられる。１つのノード１０１は、１つのファイル１０５について、read権の時刻トークンとwrite権の時刻トークンを同時に取得できる。ただし、或るノード１０１内のクライアント部１０２がサーバ部１０３に或るファイル１０５についてのread権の時刻トークンを要求したときに、他のノード１０１内のクライアント部１０２がそのファイル１０５についてのwrite 権の時刻トークンを持っていた場合には、サーバ部１０３は、その、他のノード１０１内の時刻トークンを回収する。また逆に、或るノード１０１内のクライアント部１０２がサーバ部１０３に或るファイル１０５についてのwrite 権の時刻トークンを要求したときに、他のノード１０１内のクライアント部１０２がそのファイル１０５についてのread権の時刻トークンを持っていた場合も、サーバ部１０３は、その、他のノード１０１内の時刻トークンを取り上げる。すなわち、１つのファイル１０５については、複数のノードがそれぞれ、そのファイル１０５についてのread権の時刻トークンとwrite 権の時刻トークンを同時に保有するということはない。

３）ファイル１０５ごとに複数種類（例えば４種類）のトークンが用意され、その中に、ファイルサイズの縮小を制御しmultiple-read/single-write特性を有する属性トークンが含めさせられる。

４）ファイル１０５ごとに複数種類（例えば４種類）のトークンが用意され、その中に、ファイル内データのアクセス権を制御しファイル１０５を構成するブロックごとに存在するmultiple-read/single-write特性を有する

データトークンが含めさせられる。また、本実施の形態は、下記の基本的動作を実行する。
５）各トークンは、サーバ部１０３によって管理され、トークンが必要なノード１０１内のクライアント部１０２は、サーバ部１０３に、必要なトークンの獲得を要求（依頼）する。
６）サーバ部１０３は、ファイル１０５を格納するディスク上のどこが空いているかを示す空きブロック情報（空きエクステント情報）及び個々のファイル１０５のディスク上での存在場所（ファイル１０５のエクステント情報）を、メタデータ１０４として管理している。
７）クライアント部１０２は、サーバ部１０３に、ディスク上の空きブロック群（空きエクステント群）を事前要求（リザーブ要求）し、ユーザプログラム２０２からのwrite 要求時には、事前要求で確保しておいた空きエクステント群の中から最適なものを割り当て、そこにユーザデータを書き込む。
続いて、本実施の形態の具体的な動作について、以下に順次説明する。
【００４４】図３は、任意のノード１０１内のクライアント部１０２が実行するファイル操作要求制御のメイン動作フローチャートであり、図５及び図６は、主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）が実行するファイル操作要求制御のメイン動作フローチャートである。なお、以下の説明において、特に言及しない場合には、「サーバ部１０３」と記述した場合には、主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）を指すものとする。
１）クライアント部１０２及びサーバ部１０３でのopen操作処理
任意のノード１０１において、ユーザプログラム２０２（図２）がファイル１０５のopen要求を実行すると、同一のノード１０１内のクライアント部１０２がそのopen要求を受け取る（図３のステップ３０１の判定がＹＥＳ）。この結果、クライアント部１０２は、open操作処理を実行する（図３のステップ３０２）。図４は、クライアント部１０２が実行する図３のステップ３０２のopen操作処理の動作フローチャートである。
【００４５】まず、クライアント部１０２は、ＬＡＮ１０６（図１）を介して、サーバ部１０３に、open要求を送信する。このopen要求には、アクセスの種別を示すオープンモード（read又はwrite ）が付加される。
【００４６】その後、クライアント部１０２は、サーバ部１０３からの応答を待つ（図４のステップ４０２－＞４０３－＞４０２の処理ループ）。なお、タイムアウト時には、クライアント部１０２は、エラー処理を実行し（図４のステップ４０３－＞４０４）、その後、図３のメイン動作フローチャートの処理ループに戻る。
【００４７】サーバ部１０３は、クライアント部１０２からopen要求を受信すると（図５のステップ５００の判定がＹＥＳ）、open操作処理を実行する（図５のステップ５０１）。図７は、サーバ部１０３が実行する図５のステップ５０１のopen操作処理の動作フローチャートである。
【００４８】まず、サーバ部１０３は、受信されたopen要求によって指定されているファイル１０５（図１）について、そのopen要求によって指定されているオープンモードと矛盾するデータトークンを他のノード１０１に渡しているかどうかを調べる（図７のステップ７０１）。
【００４９】サーバ部１０３は、上記オープンモードと矛盾するデータトークンを他のノード１０１に渡していない場合に、ファイル全体のデータトークン及びエクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データを、それぞれ応答データとして設定し（図７のステップ７０２～７０６）、応答処理を実行する（図７のステップ７０７）。ファイル全体のデータトークンとサイズトークンは、それぞれ、前記open要求によって指定されているオープンモードが、readならread権のトークン、write ならwrite 権のトークンである。また、時刻トークンは、write 権のトークンである。さらに、属性データには、例えばファイルサイズ、アクセス権、ファイル作成日付、ファイル更新日付等のデータが含まれる。
【００５０】一方、サーバ部１０３は、上記オープンモードと矛盾するデータトークンを他のノード１０１に渡している場合には、ファイル全体のデータトークンは設定せずに、エクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データのみを、それぞれ応答データとして設定し（図７のステップ７０３～７０６）、応答処理を実行する（図７のステップ７０７）。
【００５１】クライアント部１０２は、サーバ部１０３から応答を受信すると、その応答に含まれているファイル全体のデータトークン及びエクステント情報と、属性トークンと、サイズトークンと、時刻トークンと、属性データを、それぞれメモリ内のキャッシュ領域に保持する（図４のステップ４０２－＞４０５～４０９）。その後、クライアント部１０２は、ユーザプログラム２０２へのファイルディスクリプタの応答等の、その他のopen操作処理を実行し、その後、図３のメイン動作フローチャートの処理ループに戻る。
【００５２】以上のようにして、本実施の形態では、ファイル１０５のopen時に、競合が発生していなければ、以降のファイルアクセス（readアクセス又はwrite アクセス）に必要なトークンが全て渡されるため、クライアント部１０２は、サーバ部１０３との間で、トークン獲得のための通信を行う必要が全くなくなるという効果を有する。
【００５３】また、open要求時にファイル全体のトーク

ンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、１つのノード１０１からのwrite 要求の発行時に他のノード１０１からread命令が発行される確率は小さい。従って、１つのノード１０１に引き渡されたファイル全体のトークンが回収される確率も低く、ファイル１０５への連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

２）クライアント部１０２でのread操作処理

任意のノード１０１で、ユーザプログラム２０２がファイル１０５のread要求を発行すると、同一のノード１０１内のクライアント部１０２がそのread要求を受け取る（図３のステップ３０３の判定がＹＥＳ）。この結果、クライアント部１０２は、read操作処理を実行する（図３のステップ３０４）。図８は、クライアント部１０２が実行する図３のステップ３０４のread操作処理の動作フローチャートである。

【００５４】まず、クライアント部１０２は、必要な以下のトークンを保持しているかどうかを調べる（図８のステップ８０１）。

・read要求された範囲のread権のデータトークン
・属性トークン
・write 権の時刻トークン
・read要求が最終ブロックのread要求である場合のみ、その最終ブロックについてのread権のサイズトークン

ここで、属性トークンが存在すれば、ファイル１０５の最終ブロックの１つ前のブロックまではファイル内容が変更されていないことが保証されるため、かかるブロックのread操作処理時にはサイズトークンは獲得する必要はない。一方、read要求が最終ブロックのread要求である場合において、上記サイズトークンが存在しない場合には、他のノード１０１内のクライアント部１０２がその最終ブロックからのファイルサイズの拡張処理（write 操作処理）を実行している可能性があり、最終ブロックのread可能範囲が保証されない。上記サイズトークンが獲得された場合には、最終ブロックのread可能範囲が保証されるため、ユーザプログラム２０２は、その最終ブロックについてのread操作処理が可能となる。

【００５５】このように本実施の形態では、ファイル１０５の最終ブロックにアクセスするのでなければ、サイズトークンを獲得することなくファイル１０５にアクセスすることが可能となり、これと並行して、他のノード１０１は、サイズトークンを獲得してファイル１０５の最終ブロックにアクセスし、ファイル１０５のサイズを拡張するwrite 操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノード１０１で同時に実行させることが可能となり、システム全体の性能を向上させることができる。

【００５６】クライアント部１０２は、もし上記トークンを全て保持しているなら、サーバ部１０３にトークンを要求することなく、クライアント部１０２が保持する（キャッシュしている）データを使って、ユーザプログラム２０２の要求を処理する（図８のステップ８０１－＞８０２）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

【００５７】一方、クライアント部１０２は、もし不足するトークンが存在するなら、そのトークンをＬＡＮ１０６（図１）を介してサーバ部１０３に要求し、サーバ部１０３からの応答を待つ（図８のステップ８０１－＞８０３，ステップ８０４－＞８０５－＞８０４の処理ループ）。なお、タイムアウト時には、クライアント部１０２は、エラー処理を実行し（図４のステップ４０３－＞４０４）、その後、図３のメイン動作フローチャートの処理ループに戻る。

【００５８】クライアント部１０２は、サーバ部１０３から応答を受信すると、その応答に基づいてユーザプログラム２０２の要求を処理する（図８のステップ８０４－＞８０７）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

３）クライアント部１０２でのwrite 操作処理

任意のノード１０１で、ユーザプログラム２０２がファイル１０５のwrite 要求を発行すると、同一のノード１０１内のクライアント部１０２がそのwrite 要求を受け取る（図３のステップ３０５の判定がＹＥＳ）。この結果、クライアント部１０２は、write 操作処理を実行する（図３のステップ３０６）。この処理は、read操作処理と同様の図８の動作フローチャートによって示される。

【００５９】まず、クライアント部１０２は、必要な以下のトークンを保持しているかどうかを調べる（図８のステップ８０１）。

・write 要求された範囲のwrite 権のデータトークン
・属性トークン
・write 権の時刻トークン
・write 要求が最終ブロックのwrite 要求である場合のみ、その最終ブロックについてのwrite 権のサイズトークン

ここで、サイズトークンを用いることにより得られる効果は、read操作処理時の場合と同様である。

【００６０】クライアント部１０２は、もし上記トークンを全て保持しているなら、サーバ部１０３にトークンを要求することなく、クライアント部１０２が保持する（キャッシュしている）データを使って、ユーザプログラム２０２の要求を処理する（図８のステップ８０１－＞８０２）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

【００６１】一方、クライアント部１０２は、もし不足

するトークンが存在するなら、そのトークンをＬＡＮ１０６（図１）を介してサーバ部１０３に要求し、サーバ部１０３からの応答を待つ（図８のステップ８０１－＞８０３，ステップ８０４－＞８０５－＞８０４の処理ループ）。なお、タイムアウト時には、クライアント部１０２は、エラー処理を実行し（図４のステップ４０３－＞４０４）、その後、図３のメイン動作フローチャートの処理ループに戻る。

【００６２】クライアント部１０２は、サーバ部１０３から応答を受信すると、その応答に基づいてユーザプログラム２０２の要求を処理する（図８のステップ８０４－＞８０７）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

４）クライアント部１０２でのファイル時刻操作処理

任意のノード１０１において、ユーザプログラム２０２（図２）がファイル１０５に関するファイル時刻を要求すると、同一のノード１０１内のクライアント部１０２がその要求を受け取る（図３のステップ３０７の判定がＹＥＳ）。この結果、クライアント部１０２は、ファイル時刻操作処理を実行する（図３のステップ３０８）。図９は、クライアント部１０２が実行する図３のステップ３０８のファイル時刻操作処理の動作フローチャートである。

【００６３】まず、クライアント部１０２は、ユーザプログラム２０２から指定されたファイル１０５について、read権の時刻トークンのみを保持しているかどうかを調べる（図９のステップ９０１）。この判定がＹＥＳならば、クライアント部１０２は、自身が保持するファイル時刻をユーザプログラム２０２に応答する（図９のステップ９０３）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

【００６４】上記判定がＮＯならば、クライアント部１０２は次に、ユーザプログラム２０２から指定されたファイル１０５について、read権とwrite 権の各時刻トークンを保持しており、かつ前回サーバ部１０３から上記ファイル１０５に関するファイル時刻を取得してからそのファイル１０５に未アクセスであるかどうかを調べる（図９のステップ９０２）。この判定がＹＥＳの場合にも、クライアント部１０２は、自身が保持するファイル時刻をユーザプログラム２０２に応答する（図９のステップ９０３）。その後、クライアント部１０２は、図３のメイン動作フローチャートの処理ループに戻る。

【００６５】上記ステップ９０３の判定もＮＯならば、クライアント部１０２は、ＬＡＮ１０６を介してサーバ部１０３に、自クライアント部１０２でのそのファイル１０５に関するファイルアクセスの有無を付加した要求であって、read権の時刻トークンの獲得要求を送信する（図９のステップ９０４）。

【００６６】その後、クライアント部１０２は、サーバ部１０３からの応答を待つ（図９のステップ９０５－＞９０６－＞９０５の処理ループ）。なお、タイムアウト時には、クライアント部１０２は、エラー処理を実行し（図９のステップ９０６－＞９０７）、その後、図３のメイン動作フローチャートの処理ループに戻る。

【００６７】クライアント部１０２は、サーバ部１０３からファイル時刻を受信すると、そのファイル時刻をユーザプログラム２０２に応答する（図９のステップ９０５－＞９０８）。また、クライアント部１０２は、そのファイル時刻を、クライアント部１０２内の上記ファイル１０５に対応するキャッシュ領域に保持する（図９のステップ９０９）。さらにクライアント部１０２は、上記キャッシュ領域において、上記ファイル１０５に対してファイルアクセスなしの状態を設定する（図９のステップ９１０）。

５）サーバ部１０３でのread権の時刻トークンの応答処理

任意のノード１０１において、クライアント部１０２が、前述した図３のステップ３０８及び図９のファイル時刻操作処理を実行することによって、サーバ部１０３にread権の時刻トークンを要求すると（図９のステップ９０４）、サーバ部１０３が、それを受け取ることにより（図５のステップ５０２の判定がＹＥＳ）、read権の時刻トークンの応答処理を実行する（図５のステップ５０３）。図１０は、サーバ部１０３が実行する図５のステップ５０３の応答処理の動作フローチャートである。

【００６８】サーバ部１０３は、クライアント部１０２からread権の時刻トークンの獲得要求を受信すると、まずその時刻トークンに対応するwrite 権の時刻トークンを保持するクライアント部１０２が存在するかどうかを調べる（図１０のステップ１００１）。

【００６９】この判定がＹＥＳの場合は、クライアント部１０２は、上記write 権の時刻トークンを保持する全てのクライアント部１０２に、そのwrite 権の時刻トークンの回収要求を発行し、全てのクライアント部１０２からの応答を待つ（図１０のステップ１００１－＞１００２，ステップ１００３－＞１００４－＞１００３の処理ループ）。なお、タイムアウト時には、サーバ部１０３は、エラー処理を実行し（図１０のステップ１００４－＞１００５）、その後、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【００７０】これに対して、各クライアント部１０２では、要求されたwrite 権の時刻トークンの回収処理を実行する（図３のステップ３０９－＞３１０）。具体的には、各クライアント部１０２は、要求されたwrite 権の時刻トークンを無効化すると共に、その時刻トークンに対応するファイル１０５に対するファイルアクセスの有無を、サーバ部１０３への応答に付加する。

【００７１】サーバ部１０３は、ステップ１００１の判定がＮＯであった場合、又は上記write 権の時刻トークンを保持する全てのクライアント部１０２からの応答を

受信した場合に、read権の時刻トークンを要求しているクライアント部１０２に応答するファイル時刻を決定する（図１０のステップ１００６）。具体的には、要求元を含めて（図９のステップ９０４参照）、いずれかのノード１０１のクライアント部１０２がファイルアクセス有りを応答した場合は、サーバ部１０３は、自身がメタデータ１０４として保持する該当ファイル時刻を、現時刻により更新する。なお、各クライアント部１０２からファイルアクセス相対時刻間隔（何秒前にアクセスしたかを示すデータ）を応答させるようにし、応答された各クライアント部１０２からのファイルアクセス相対時刻間隔のうち最も小さい値によって、メタデータ１０４内の時刻を更新する（すなわち、［“現時刻”－“最も小さいファイルアクセス相対時刻間隔］にする）ように構成されてもよい。一方、いずれのノード１０１もファイルアクセス無しを応答した場合は、サーバ部１０３は、自身が保持するメタデータ１０４中の該当ファイル時刻を、そのまま使用する。

【００７２】続いて、サーバ部１０３は、決定したメタデータ１０４中のファイル時刻を、read権の時刻トークンを要求したクライアント部１０２に応答する（図１０のステップ１００７）。

【００７３】最後に、サーバ部１０３は、要求元のクライアント部１０２にread権の時刻トークンを渡したことをサーバ部１０３の主記憶中に記憶する（図１０のステップ１００８）。

【００７４】その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

６）サーバ部１０３でのwrite 権の時刻トークンの応答処理

任意のノード１０１において、クライアント部１０２が、前述した図３のステップ３０４及び図８のread操作処理又は図３のステップ３０６及び図８のwrite操作処理を実行することにより、サーバ部１０３にwrite 権の時刻トークンを要求すると、サーバ部１０３が、それを受け取ることにより（図５のステップ５０４の判定がＹＥＳ）、write 権の時刻トークンの応答処理を実行する（図５のステップ５０５）。図１１は、サーバ部１０３が実行する図５のステップ５０５の応答処理の動作フローチャートである。

【００７５】サーバ部１０３は、クライアント部１０２からwrite 権の時刻トークンの獲得要求を受信すると、まずその時刻トークンに対応するread権の時刻トークンを保持するクライアント部１０２が存在するかどうかを調べる（図１１のステップ１１０１）。

【００７６】この判定がＹＥＳの場合は、クライアント部１０２は、上記read権の時刻トークンを保持する要求クライアント部１０２を除く全てのクライアント部１０２に、そのread権の時刻トークンの回収要求を発行し、全てのクライアント部１０２からの応答を待つ（図１１のステップ１１０１－＞１１０２，ステップ１１０３－＞１１０４－＞１１０３の処理ループ）。なお、タイムアウト時には、サーバ部１０３は、エラー処理を実行し（図１１のステップ１１０４－＞１１０５）、その後、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【００７７】これに対して、各クライアント部１０２では、要求されたread権の時刻トークンの回収処理を実行する（図３のステップ３０９－＞３１０）。具体的には、各クライアント部１０２は、要求されたread権の時刻トークンを無効化し、サーバ部１０３に応答を返す。

【００７８】サーバ部１０３は、ステップ１１０１の判定がＮＯであった場合、又は上記read権の時刻トークンを保持する全てのクライアント部１０２からの応答を受信した場合に、write 権の時刻トークンを、要求クライアント部１０２に応答する（図１１のステップ１１０６）。

【００７９】最後に、サーバ部１０３は、要求元のクライアント部１０２にwrite 権の時刻トークンを渡したことをメタデータ１０４中に記憶する（図１１のステップ１１０７）。

【００８０】その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。上述の２）～６）で示したように、本実施の形態では、ユーザプログラム２０２がファイル１０５のread操作処理又はwrite 操作処理を実行するときには、該当クライアント部１０２はそのファイル１０５についてのwrite 権の時刻トークンを使用する。この際、クライアント部１０２はそのファイル１０５についてのwrite 権の時刻トークンを保持していなければサーバ部１０３にそれを要求する。これに応答してサーバ部１０３は、他のノード１０１からそのファイル１０５に対応するread権の時刻トークンは回収するが、write 権の時刻トークンは回収しない。従って、クライアント部１０２は、ユーザプログラム２０２が１つのファイル１０５に連続アクセスするような場合において、そのファイル１０５への最終的なアクセスが終了するまでwrite 権の時刻トークンを返却する必要も、またアクセスの有無をサーバ部１０３に通知する必要もなく、他のノード１０１との間でそのファイル１０５のファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能となる。

【００８１】なお、上述の制御によると、write 権の時刻トークンは、ユーザプログラム２０２がファイル１０５のファイル時刻を明示的に要求し、該当クライアント部１０２からサーバ部１０３にそのファイル１０５についてのread権の時刻トークンが要求された場合に回収されることになるが、これだけだと、ファイル時刻の要求が発生しない限り、ファイル１０５のファイル時刻がいつまでたってもサーバ部１０３側で確定しないことにな

る。これを防ぐために、例えば、クライアント部１０２は、ユーザプログラム２０２がファイル１０５をクローズしたタイミングで、サーバ部１０３にファイルアクセスの有無を通知し、サーバ部１０３はそれを受けてメタデータ１０４中の該当ファイル時刻を更新するように構成することができる。

７）サーバ部１０３でのデータトークンの応答処理

任意のノード１０１において、クライアント部１０２が、前述した図３のステップ３０４及び図８のread操作処理又は図３のステップ３０６及び図８のwrite操作処理を実行することにより、サーバ部１０３にデータトークンを要求すると（図８のステップ８０３）、サーバ部１０３が、それを受け取ることにより（図５のステップ５０６の判定がＹＥＳ）、データトークンの応答処理を実行する（図５のステップ５０７）。図１２は、サーバ部１０３が実行する図５のステップ５０７の応答処理の動作フローチャートである。

【００８２】サーバ部１０３は、クライアント部１０２からデータトークンの獲得要求を受信すると、まずその要求に矛盾するデータトークンを保持するクライアント部１０２が存在するかどうかを調べる（図１２のステップ１２０１）。

【００８３】この判定がＹＥＳの場合は、クライアント部１０２は、上記データトークンを保持する全てのクライアント部１０２に、そのデータトークンの回収要求を発行し、全てのクライアント部１０２からの応答を待つ（図１２のステップ１２０１－＞１２０２，ステップ１２０３－＞１２０４－＞１２０３の処理ループ）。なお、タイムアウト時には、サーバ部１０３は、エラー処理を実行し（図１２のステップ１２０４－＞１２０５）、その後、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【００８４】これに対して、各クライアント部１０２では、要求されたデータトークンの回収処理を実行する（図３のステップ３０９－＞３１０）。具体的には、各クライアント部１０２は、要求されたデータトークンを無効化し、サーバ部１０３に応答を返す。また、回収を要求されたデータトークンがwrite 権のデータトークンである場合には、各クライアント部１０２は、そのwrite 権のデータトークンで示されるファイル１０５の範囲で自身が更新したデータをキャッシュからディスク上に書き戻し、新たにそのファイル１０５に割り当てたエクステント情報を、上記応答に付加する。

【００８５】サーバ部１０３は、上述のデータトークンを保持する全てのクライアント部１０２からの応答を受信した場合に、上記応答がwrite 権のデータトークンに関するものであるならば、応答されたファイル１０５のエクステント情報を、自身が保持するメタデータ１０４に反映させる（図１２のステップ１２０３－＞１２０６）。

【００８６】その後、サーバ部１０３は、要求元のクライアント部１０２から指定された範囲のエクステント情報が付加されたデータトークンを、上記クライアント部１０２に応答する（図１２のステップ１２０７）。

【００８７】一方、クライアント部１０２からのデータトークンの獲得要求に矛盾するデータトークンを保持するクライアント部１０２が存在せずステップ１２０１の判定がＮＯで、かつファイル全体のデータトークンを応答しても競合が発生せずステップ１２０８の判定もＮＯである場合には、サーバ部１０３は、ファイル全体のエクステント情報とファイル全体のデータトークンを、要求元のクライアント部１０２に応答する（図１２のステップ１２０１－＞１２０８－＞１２０９）。

【００８８】上記競合が発生する場合には、サーバ部１０３は、要求元のクライアント部１０２から指定された範囲のエクステント情報が付加されたデータトークンを、上記クライアント部１０２に応答する（図１２のステップ１２０７）。

【００８９】ステップ１２０７又は１２０９の処理の後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。サーバ部１０３からデータトークンを取得したクライアント部１０２は、前述した図４のステップ４０５又は図８のステップ８０７の処理において、自身が該当ファイル１０５に対応するデータトークンを保持していること、及び応答されたエクステント情報を、メモリ内のキャッシュ領域に記憶する。そして、クライアント部１０２は、それ以降のユーザプログラム２０２からの要求に基づくファイルアクセス処理（図８のステップ８０２）は、上記エクステント情報で示される、ディスク上のブロックに対して実行する。

【００９０】上述したように、データトークンの応答時に、ファイル１０５のエクステント情報も同時に応答される。このため、複数のノード１０１は、ディスク装置内のファイル１０５に、ＬＡＮ１０６経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

８）サーバ部１０３におけるサイズトークンの応答処理

サーバ部１０３は、クライアント部１０２からサイズトークンを要求された場合には、その要求と矛盾するサイズトークンを他のクライアント部１０２から回収した上で、要求されたサイズトークンにファイルサイズを付加して要求元のクライアント部１０２に応答する（図５のステップ５０６－＞５０７）。その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

９）サーバ部１０３における属性トークンの応答処理

サーバ部１０３は、クライアント部１０２から属性トークンを要求された場合には、その要求と矛盾する属性トークンを他のクライアント部１０２から回収した上で、

要求された属性トークンにファイル属性を付加して要求元のクライアント部１０２に応答する（図５のステップ５０８－＞５０９）。その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

１０）エクステント管理の詳細

次に、サーバ部１０３及びクライアント部１０２におけるエクステント（ディスク領域）の管理の詳細について説明する。

【００９１】まず、サーバ部１０３は、複数のディスクボリュームを管理することができ、メタデータ１０４として、ファイル１０５の属性データ、各ディスクボリューム毎の空きエクステントに関する情報（空きスペース情報）、及びクライアント部１０２に貸し出したエクステントに関する情報（リザーブスペース情報）を保持している。

【００９２】空きスペース情報とリザーブスペース情報は、図１３に示されるように、空きスペースＢツリー１３０１として管理され、そのうち空きスペース情報は空きスペースキュー１３０２からアクセスでき、リザーブスペース情報はリザーブスペースキュー１３０３からアクセスできる。

【００９３】空きスペースキュー１３０２は、ディスクボリューム毎に、空きスペースＢツリー１３０１に接続されている使用可能エクステント（使用中でもリザーブ中でもないエクステント）を管理する。

【００９４】リザーブスペースキュー１３０３は、クライアント部１０２毎に、そのクライアント部１０２にリザーブされ空きスペースＢツリー１３０１に接続されているエクステントを管理する。

【００９５】また、サーバ部１０３は、使用中のエクステントは、ｉノードＢツリー１３０４によって管理する。一方、クライアント部１０２は、サーバ部１０３に要求することによりリザーブしたエクステントを、リザーブキュー１３０５によって管理する。

【００９６】クライアント部１０２は、主記憶上にキャッシュを持ち、ユーザプログラムが要求したディスク上のデータをキャッシュする。サーバ部１０３内の空きスペーアスキュー１３０２とクライアント部１０２内のリザーブキュー１３０５は、ディスクボリューム毎に予め決められた個数分のヘッダを有しており、各ヘッダがエクステントのサイズに対応している。例えば、ヘッダの個数を４個とすると、各ヘッダが、１～４ＫＢ（キロバイト）、４～１６ＫＢ、１６～６４ＫＢ、６４～２５６ＫＢの各サイズ範囲のエクステント群（空きスペースＢツリー１３０１）を管理する。ヘッダの個数と各ヘッダが表すサイズは、各ディスクボリュームのファイルシステムを作成したときに決定される。

【００９７】図１４は、１つのノード１０１（図１参照）内において、ユーザプログラム２０２（図２参照）が、ファイル１０５へのデータ書き込み（write 要求）を依頼したときのエクステント管理のシーケンスを示す図である。このシーケンスにおいて、クライアント部１０２が実行する処理は、図３のステップ３０６のwrite操作処理における図８のステップ８０７の処理の一部である。また、サーバ部１０３が実行する処理は、図５のサーバ部１０３のメイン動作フローチャート内の特には図示しない一部の処理である。

【００９８】図１４において、ユーザプログラム２０２がファイル１０５に対するwrite 要求を発行すると、クライアント部１０２は、キャッシュにデータを保持する。ユーザプログラム２０２がファイル１０５をクローズし、又はキャッシュが一杯になり、或いはサーバ部１０３からデータトークンの回収を要求される（図１２のステップ１２０２参照）ことにより、キャッシュされているデータをディスクに書き出す必要が発生した場合に、クライアント部１０２は、サーバ部１０３から受け取っていたファイル１０５のエクステント情報（図４のステップ４０５参照）を調べ、その要求が既にディスク領域が割り当てられているファイル領域に対するものであるか否かを認識し、ファイル１０５毎にキャッシュ内でエクステントが割り当てられていない領域で隣接するものをまとめる（このまとめられたファイル領域を書出し対象領域と呼ぶ）。次に、クライアント部１０２は、書出し対象領域のサイズを調べると共に、その領域の性質に従って、以下の何れかの処理を実行する。

①書出し対象領域に隣接する（直前の）領域に、同じファイル１０５に関するエクステントが既にサーバ部１０３から割り当てられている場合：クライアント部１０２は、割り当てられているエクステントのブロックアドレスと書出し対象領域のサイズを指定して、それに続くエクステントのリザーブ（貸し出し）をサーバ部１０３に依頼し、応答されたエクステントにデータを書き込む。なお、サーバ部１０３は、依頼されたエクステントが既に割当て済みの場合には、他のエクステントを返す。

②書出し対象領域に隣接する（直前の）領域に、同じファイル１０５に関するエクステントがいまだサーバ部１０３から割り当てられていない場合：クライアント部１０２は、書出し対象領域のサイズに対応するリザーブキュー１３０５の先頭に接続されているエクステントにデータを書き出す。クライアント部１０２は、リザーブキュー１３０５から、そのエクステントを取り除く。

以上の動作の後、クライアント部１０２は、サーバ部１０３に書出し完了を通知する。この際、クライアント部１０２は、使用したエクステント（リザーブスペース）のアドレスと、書出し対象領域のサイズを通知する。

【００９９】サーバ部１０３は、通知されたエクステント（リザーブスペース）のアドレスと、書出し対象領域のサイズとから、メタデータ１０４内の対象ファイル１０５に関する属性データを更新し、リザーブスペースキ

ュー１３０３及び空きスペースＢツリー１３０１上から、クライアント部１０２から通知されたエクステントを取り除き、そのエクステントをＩノードＢツリー１３０４に接続する。書き出されたエクステントのサイズが使用されたリザーブスペースよりも小さい場合には、サーバ部１０３は、残りのエクステントを、空きスペースとして空きスペースキュー１３０２の当該エクステントのサイズに対応するヘッダに接続する。

１１）エクステント群のリザーブ制御処理

クライアント部１０２は、一定時間が経過するごとに、エクステント群リザーブ要求処理を実行する（図３のステップ３１１－＞３１２）。この処理では、クライアント部１０２は、自身がリザーブキュー１３０５にリザーブしているエクステント群を調べ、リザーブ量が一定値以下になった場合に、サーバ部１０３に一定個数のエクステント群のリザーブを要求する。この処理は、各サイズのヘッダ毎に行われ、不足が発生したヘッダ以外についても、各リザーブ量が所定値以上となるように、各ヘッダに対して上記リザーブ処理が実行される。

【０１００】サーバ部１０３は、エクステント群のリザーブ要求を受信すると、エクステント群のリザーブ処理を実行する（図６のステップ５１２－＞５１３）。この処理では、サーバ部１０３は、空きスペースキュー１３０２に接続されている空きスペースＢツリー１３０１中から、使用可能なエクステント群を探し、それらを空きスペースキュー１３０２からリザーブスペースキュー１３０３に繋ぎ替えた後に、そのリザーブしたエクステント群をクライアント部１０２に応答する。その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【０１０１】クライアント部１０２は、図３のステップ３１２において、サーバ部１０３から応答されたエクステント群をリザーブキュー１３０５に繋ぎ、ステップ３１２を終了して、図３のメイン動作フローチャートの処理ループに戻る。

【０１０２】サーバ部１０３は、自身に対してmount を行っているクライアント部１０２の障害を検出した場合、又はクライアント部１０２からunmount 要求を受信した場合には、そのクライアント部１０２に対してリザーブしていたリザーブスペースキュー１３０３中のエクステント群の解放処理を実行して、それらを空きスペースキュー１３０２に繋ぎ替える（図５のステップ５１４－＞５１５）。その後、サーバ部１０３は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【０１０３】上述のように、本実施の形態では、空きエクステント群がリザーブされることにより、クライアント部１０２は、サーバ部１０３に問い合わせることなく、キャッシュを活用して新たなエクステントをファイル１０５に割り当てることが可能となる。このため、クライアント部１０２とサーバ部１０３との間の通信回数を削減でき、システム全体の性能を向上させることが可能となる。

【０１０４】また、新たに割り当てられたエクステントは、データが書き込まれた後のクライアント部１０２からサーバ部１０３への応答によって初めて、そのファイル１０５のメタデータ１０４として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

１２）主サーバと従サーバの同期処理

主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）は、例えば図７、図１０、図１１、図１２などにおいて、メタデータ１０４（＃１）を更新する場合は、従サーバであるノード１０１（＃２）内のサーバ部１０３（＃２）に対して、メタデータ変更分と時刻データを送信し、従サーバがそれらを受信したことを確認した後に、クライアント部１０２に応答を返す。

【０１０５】従サーバであるノード１０１（＃２）内のサーバ部１０３（＃２）は、上述のメタデータ変更分と時刻データを受信すると、メタデータ変更分を自身のメタデータ１０４（＃２）に反映させると共に、送られてきた時刻データを記憶する（図６のステップ５１６－＞５１７）。その後、サーバ部１０３（＃２）は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

１３）主サーバにおける障害発生時の、従サーバへの切替処理

従サーバであるノード１０１（＃２）内のサーバ部１０３（＃２）は、主サーバであるノード１０１（＃１）内のサーバ部１０３（＃１）の障害を監視しており、その障害を検出した場合には、サーバ切替処理を実行する（図６のステップ５１８－＞５１９）。このとき、サーバ部１０３（＃２）は、最後に主サーバであるサーバ部１０３（＃１）から送られてきた時刻を過ぎるまで、自身の時刻の待ち合せを実行する。その後、サーバ部１０３（＃２）は、図５及び図６のメイン動作フローチャートの処理ループに戻る。

【０１０６】上述の制御により、サーバ切替時にも、矛盾のないファイル時刻の付与が可能となる。次に、上述したようなノード間ファイル共用管理システムにおいて、分散ファイルシステムの耐故障性を高めるためのログ制御機構を実現するための実施の形態について説明する。

【０１０７】図１５は、ログ制御機構を実装したノード間ファイル共用管理システムの基本構成図である。共用ファイル管理装置１５０１（図１のサーバ部１０３を有するノード１０１に対応する）は、共用されるファイルの「属性」や「実ディスク上での格納位置」などの、ファイルごとに存在する制御情報（ファイル情報と呼ぶ）と、実ディスクの空き領域などを示す制御情報（ディスク情報と呼ぶ）を保持している。これら２つの管理情報

を総称してメタデータ１５０２（図１のメタデータ１０４に対応する）と呼び、障害に備えディスク上に格納されている。

【０１０８】共用ファイル管理装置１５０１は、データを共用する＃１～＃ｎの各ノード１５０３（クライアント部１０２を有するノード１０１に対応する）からの要求に従い、メタデータ１５０２をディスクから読み込み或いは更新し、ファイル情報を応答として返す。この際、異なる複数のメタデータブロックがアクセスされる可能性がある。

【０１０９】各ノード１５０３は、返されたファイル情報をメモリ上にキャッシュし、それ以降必要が生ずるまで、共用ファイル管理装置１５０１と通信することなく、キャッシュされたメモリ上のファイル情報のみを用いて処理を実行する。

【０１１０】各ノード１５０３がそれぞれのキャッシュ上に保持するファイル情報相互間の一貫性を保証するために、トークンが使用される。トークンは、ファイル情報がノード１５０３に返される際に共用ファイル管理装置１５０１によりそのノード１５０３に対して発行され、共用ファイル管理装置１５０１が或るノード１５０３から矛盾する要求を受け付けたときに共用ファイル管理装置１５０１によって必要なノード１５０３から回収される。

【０１１１】回吸を指示されたノード１５０３は、トークンによって指示されるキャッシュデータを無効化し、他ノード１５０３に伝えられるべき自身が行なったファイル情報の変更を応答する。

【０１１２】応答を受けた共用ファイル管理装置１５０１は、通知された変更をメタデータ１５０２に反映した後に、要求に基づく処理を再開し、要求元に対して結果を応答すると共にトークンを発行する。

【０１１３】共用ファイル管理装置１５０１が各ノード１５０３からの要求を処理するためには、メタデータ１５０２へのアクセスが必要となる。この場合に、毎回ディスクをアクセスしていたのでは性能が悪くなる。このため、ディスク上のデータを保持するバッファキャッシュ１５０４が共用ファイル管理装置１５０１内に設けられ、ディスクアクセスが削減される。バッファキャッシュ１５０４は、ディスク上の各ブロックに対応したエントリを持ち、各エントリにそのエントリのロックの有無を表示するためのロックワードが用意されることにより、或るスレッドが更新中のデータを他の要求を処理している他のスレッドが参照することが抑止される。

【０１１４】メタデータ１５０２の実ディスクへの反映は、要求処理が全て正常に終了した時点、いわゆるトランザクション完了時まで遅らされる。トランザクションが正常に終了すると、バッファキャッシュ１５０４上に保持されている更新データが一括してログファイル１５０５に書き出され、その後、更新データのディスクへの反映タイミングがスケジュールされる。

【０１１５】ログファイル１５０５はサイクリックに使用され、実ディスクへの書き込みが完了するたびに、書出しが完了した変更を保持するログ領域は空き領域に戻される。従って、実ディスクへの書出しがまだ完了していない、成功した要求に伴うメタデータの変更は必ずログファイル１５０５上に存在するので、共用ファイル管理装置１５０１で障害が発生しても、メタデータ１５０２の復旧は容易にかつ高速に行なえるという特徴を有する。

【０１１６】次に、本実施の形態に係る上記基本構成に基づくロック継承制御処理につき、図１６の説明図に基づいて説明する。尚、複数のクライアントから発行れる同一ファイルに対する操作要求を逐次化するためのファイル管理装置１５０１はファイル毎に用意するファイルロックを使用する。

【０１１７】本実施の形態では、１つのノード１５０３からの要求を処理するために共用ファイル管理装置１５０１上で実行される第１の実行単位（スレッド）は、他のノード１５０３に発行しているトークンを回収する場合に、トークン処理の対象となっているファイルを示す情報を保持したトークン回収制御表１６０２をトークン回収待ちキュー１６０１につなぎ、該当するノード１５０３に対してトークン回収要求を送信した後、トークン回収完了メッセージの到着を待ち合わせる。

【０１１８】トークンを保持しているノード１５０３におけるキャッシュの無効化が完了しそこから共用ファイル管理装置１５０１（図１５）にトークン回収完了メッセージが通知されると、トークン回収完了メッセージを処理するために共用ファイル管理装置１５０１上で実行される第２の実行単位（スレッド）が、トークン回収待ちキュー１６０１を調べ、そのメッセージに対応するトークン回収制御表１６０２がキュー上に存在するならば、その制御表に「ロック継承中」を表示した上で、メタデータ１５０２（図１５）の更新処理及びトークンの解放処理を実行する。

【０１１９】トークン回収完了メッセージの到着を待ち合わせていた第１の実行単位の待ちは、第２の実行単位によるトークン解放処理の結果解かれる。各ノード１５０３は、共用ファイル管理装置１５０１からの要求に基づかずに自律的に、トークン回収完了メッセージを共用ファイル管理装置１５０１に通知することもできる。従って、トークン回収完了メッセージが共用ファイル管理装置１５０１に到着した際に、トークン回収待ちキュー１６０１に該当するトークン回収制御表１６０２がつながっていない場合が起こり得る。このようなときには、上記第２の実行単位は、通常のファイルロック獲得処理を実行し、この結果他の実行単位がファイルロックを保持していればファイルロックの解放を待ち合わせ、ファイルロックがはずれたらメタデータの更新処理及びトー

クン解放処理を実行する。
【０１２０】上記第１の実行単位は、複数のノード１５０３に対してトークン回収要求を送信する可能性がある。このような場合には、共用ファイル管理装置１５０１は、複数のノード１５０３からトークン回収完了メッセージを相次いで受信する可能性がある。上記第２の実行単位は、第１番目のトークン回収完了メッセージを受信した時点で該当するトークン回収制御表１６０２にロック継承中を表示する。そして、第２番目以降のトークン回収完了メッセージを受信した他の各実行単位は、対応するトークン回収制御表１６０２にロック継承中が表示されていた場合には、継承中表示がオフとなるのを待ち合わせ、待ちが解けた時点でメタデータの更新処理及びトークン解放処理を実行する。このように、ロックの継承を行うことのできる実行単位は１つに制限される。
【０１２１】以上のロック継承制御により、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。次に、本実施の形態に係る図１５に示される基本構成に基づくデッドロック検出処理について、図１７の説明図に基づき説明する。
【０１２２】共用ファイル管理装置１５０１（図１５）は、各ファイルを管理するファイル制御表１７０１に、ファイルロックワード１７０１ａに対応して、そのファイルロックを保持している実行単位（スレッド）を示すオーナ１７０１ｂを設定し、また、各バッファキャッシュ１５０４（図１５）のエントリを管理するバッファキャッシュ制御表１７０２に、バッファキャッシュロックワード１７０２ａに対応して、そのバッファキャッシュロックを保持している実行単位（スレッド）を示すオーナ１７０２ｂを設定する。
【０１２３】また、共用ファイル管理装置１５０１は、各実行単位（スレッド）を管理するスレッド制御表１７０３に、その実行単位が待ち合わせしている対象を特定する情報である待ちリソース１７０３ａと、その待ち合わせの原因を示す情報であるタイプ１７０３ｂを設定する。待ちリソース１７０３ａとタイプ１７０３ｂには下記の何れかの設定が行われる。
１．ファイルロックの解放を待ち合わせる場合：
・タイプ１７０３ｂには、ファイルロック待ちを設定。
【０１２４】・待ちリソース１７０３ａには、該当するファイル制御表１７０１内のファイルロックワード１７０１ａを指示する情報を設定。
２．バッファキャッシュロックの解放を待ち合わせる場合：
・タイプ１７０３ｂには、バッファキャッシュロック待ちを設定。
【０１２５】・待ちリソース１７０３ａには、該当するバッファキャッシュ制御表１７０２内のバッファキャッシュロックワード１７０２ａを指示する情報を設定。
３．トークン回収を待ち合わせる場合：
・タイプ１７０３ｂには、トークン回収待ちを設定。
【０１２６】・待ちリソース１７０３ａには、該当するファイルを指示する情報を設定。
以上の情報を使い、各スレッド（実行単位）は、以下のようにデッドロックを検出する。
＜スレッド（以下、スレッドＡという）がファイルロックを要求した場合＞
ステップ１：スレッドＡは、ファイルロックの解放待ちに入る前に、そのファイルに対応するファイル制御表１７０１内のファイルロックワード１７０１ａとオーナ１７０１ａとから、そのファイルロックを保持しているスレッド（以下、スレッドＢという）に対応するスレッド制御表１７０３を取得する。
ステップ２：スレッドＡは、そのスレッド制御表１７０３内の待ちリソース１７０３ａとタイプ１７０３ｂとから、スレッドＢが待ち合わせている資源を求める。スレッドＢが待ち合わせている資源がないかスレッドＢがトークン回収を待ち合わせているならば、スレッドＡは、デッドロックは発生していないと判定し、ファイルロックの解放待ちに入る。
ステップ３：スレッドＢがトークン回収の待ち合わせ以外の待ち合わせをしている場合には、スレッドＡは、スレッドＢが待ち合わせている資源に対するロックを保持しているスレッドを求める。
ステップ４：スレッドＡは、ステップ３で求めたスレッドがスレッドＡ自身ならば、デッドロックが発生したと判定し、スレッドＡ自身が実行しているトランザクションをキャンセルする。そうでなければ、スレッドＡは、ステップ２の処理を繰り返す。
＜スレッドＡがバッファキャッシュロックを要求した場合＞
ステップ１：スレッドＡは、バッファキャッシュロックの解放待ちに入る前に、そのバッファキャッシュエントリに対応するバッファキャッシュ制御表１７０２内のバッファキャッシュロックワード１７０２ａとオーナ１７０２ｂとから、そのバッファキャッシュロックを保持しているスレッドＢに対応するスレッド制御表１７０３を取得する。
ステップ２：スレッドＡは、そのスレッド制御表１７０３内の待ちリソース１７０３ａとタイプ１７０３ｂとから、スレッドＢが待ち合わせている資源を求める。スレッドＢが待ち合わせている資源がないならば、スレッドＡは、デッドロックは発生していないと判定し、バッファキャッシュロックの解放待ちに入る。
ステップ３：スレッドＡは、スレッドＢが待ち合わせている資源がトークン回収待ちという資源で且つトークン回収待対象ファイルのファイルロックをスレッドＡが保持しているならば、デッドロックが発生したと判定する。

ステップ４：スレッドＡは、スレッドＢが待ち合わせている資源に対するロックを保持しているスレッドを求める。
ステップ５：スレッドＡは、ステップ４で求めたスレッドがスレッドＡ自身ならば、デッドロックが発生したと判定し、スレッドＡ自身が実行しているトランザクションをキャンセルする。そうでなければ、スレッドＡは、ステップ２の処理を繰り返す。
以上説明したデッドロックの検出処理により、トークンに基づいてトランザクション制御されているメタデータ１５０２等の更新処理におけるデッドロックの発生を適切に検出することができる。
【０１２７】次に、本実施の形態に係る図１５に示される基本構成に基づくログファイルの２次キャッシュ制御処理につき、図１８の説明図に基づいて説明する。２次キャッシュ１８０１は、ログファイル１５０５（図１５）には書出しが完了しているが、ディスクへの反映は完了していないメタデータ１５０２を保持するキャッシュで、トランザクションキャンセル時の性能劣化の防止、通常処理での性能向上を図るために、共用ファイル管理装置１５０１上に設けられる。
【０１２８】トランザクションが正常終了した場合、バッファキャッシュ１５０４上で更新されたデータは２次キャッシュ１８０１に送られ、変更表示がオンされる。ログファイル１５０５の空き領域が不足してくると、２次キャッシュ１８０１上の変更表示がオンになっているデータが実ディスクに書き出され、変更表示がリセットされる。
【０１２９】バッファキャッシュ１５０４から２次キャッシュにデータが移動させられる際に、２次キャッシュ１８０１の空き領域がなければ、変更表示がオンされていない２次キャッシュ領域が再使用される。
【０１３０】もし、全てのページの変更表示がオンされているならば、一定の量の変更されたページが実ディスクに書き出され、変更表示がオフにさせられた後に再使用される。
【０１３１】必要なメタデータ１５０２がバッファキャッシュ１５０４上に存在しない場合には、２次キャッシュ１８０１にデータが存在するならばそのデータが２次キャッシュ１８０１からバッファキャッシュ１５０４にコピーされる。必要なデータが２次キャッシュ１８０１にも存在しない場合には、そのデータがディスクからバッファキャッシュ１５０４に読み込まれる。
【０１３２】以上説明した２次キャッシュ制御処理により、バッファキャッシュ１５０４の変更内容を実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。
【０１３３】続いて、本実施の形態に係る図１５に示される基本構成に基づく、ログデータ量を削減できるログ制御処理につき、図１９の説明図に基づいて説明する。メタデータ１５０２がバッファキャッシュ１５０４上で更新された場合に、スレッドごとに存在するログキュー１９０１に、更新されたメタデータ１５０２の範囲を示す情報を記憶したログ制御表１９０２が追加される。この情報は、図１９に示されるように、バッファキャッシュ１５０４上のエントリを指示するエントリＩＤと、そのエントリに属する範囲の始点アドレスｓｔａｒｔと終点アドレスｅｎｄとからなる。
【０１３４】この際、ログキュー１９０１がサーチされ、ログキュー１９０１上に、更新されたメタデータ１５０２の範囲に対してオーバラップするか隣接する範囲を表すログ制御表１９０２が既に存在するならば、旧制御表１９０２の範囲が変更させられるだけで、新しいログ制御表１９０２は作成されない。
【０１３５】トランザクションが正常に終了した場合、ログキュー１９０１上のログ制御表１９０２から、変更されたメタデータ１５０２が認識され、それがログファイル１５０５にログデータとして書き出される。書出しが完了したら、該当するバッファキャッシュ１５０４のエントリに対するロックが解放される。
【０１３６】トランザクションが失敗に終った場合には、ログキュー１９０１から更新されたメタデータ１５０２が認識され、該当するバッファキャッシュ１５０４上のエントリが無効化される。
【０１３７】以上説明したログ制御処理により、ログファイル１５０５に書き出されるログデータ量の削減が実現される。最後に、本実施の形態に係る図１５に示される基本構成に基づく、トランザクションキャンセル時におけるメモリ常駐制御表のリストア制御処理につき、図２０の説明図に基づいて説明する。
【０１３８】トランザクション処理の途中でデッドロック条件が検出されたり要求元のエラーなどが検出されることによりトランザクションがキャンセルされる場合には、バッファキャッシュ１５０４（図１５）の無効化が行なわれる。これと共に、スレッドごとに存在するファイルロックキュー２００１に接続されている各ファイル制御表２００２がサーチされることにより、トランザクションの過程で獲得され解放されていないファイルロックが、全て解放させられる。
【０１３９】ここで、ファイル制御表２００２には、ファイルロックの獲得に伴って、共用ファイル管理装置１５０１（図１５）内のメモリ上に存在する常駐制御表２００３が書き換えられた場合に、その更新を示す制御表更新フラグが設定される。なお、１つのファイル制御表２００２には、複数の常駐制御表２００３に対応する複数の制御表更新フラグを、制御表更新マップとして設定することができる。
【０１４０】今、トランザクションのキャンセルに伴いファイルロックが解除される際に、それに対応するファ

イルロックワードが設定されていたファイル制御表２００２において何れかの制御表更新フラグがオンになっている場合には、ファイルロックの再獲得時にその制御表更新フラグに対応する常駐制御表２００３のリロードが必要なことを示すリロードインジケータ（複数可）が表示された上で、ファイルロックが解放させられる。

【０１４１】トランザクションがデッドロック検出等によりキャンセルされた場合には、その後、そのトランザクションに対応する要求が始めからから再試行される。そして、ファイルロックの再獲得時に、それに対応するファイルロックワードが設定されていたファイル制御表２００２に何れかのリロードインジケータが表示されているならば、ファイルロックの獲得後に上記リロードインジケータによって指示される常駐制御表２００３が、メタデータ１５０２（図１５）の情報を使ってメモリ上に再構築される。

【０１４２】以上説明したリストア制御処理により、トランザクションのキャンセルに伴う常駐制御表２００３の高速なリストアが実現される。ここで、本発明は、コンピュータにより使用されたときに、上述の本発明の実施の形態によって実現されるクライアント部１０２の機能又はサーバ部１０３の機能と同様の機能をコンピュータに行わせるためのコンピュータ読出し可能記録媒体として構成することもできる。この場合に、例えばフロッピィディスク、ＣＤ－ＲＯＭディスク、光ディスク、リムーバブルハードディスク等の可搬型記録媒体や、ネットワーク回線経由で、本発明の実施の形態の各種機能を実現するプログラムが、ノードを構成するコンピュータの本体内のメモリ（ＲＡＭ又はハードディスク等）にロードされて、実行される。

【０１４３】

【発明の効果】本発明の第１の態様の構成によれば、例えばopen要求時等においてファイル全体のトークンが引き渡されることにより、可能な限り新たなトークン要求を行わずにファイルへの連続アクセスが可能となる。データベースアクセス等を除く一般的なファイルアクセスでは、１つのノードからのwrite 要求の発行時に他のノードからread命令が発行される確率は小さい。従って、１つのノードに引き渡されたファイル全体のトークンが回収される確率も低く、ファイルへの連続アクセス時にアクセス単位ごとにトークン要求が不要になることによる性能向上が期待できる。

【０１４４】本発明の第２の態様の構成によれば、クライアント装置は、ユーザプログラムが１つのファイルに連続アクセスするような場合において、そのファイルへの最終的なアクセスが終了するまでwrite 権の時刻トークンを返却する必要もまたアクセスの有無をサーバ部に通知する必要もなく、他のノードとの間でそのファイルのファイル時刻の同期をとる必要がなくなる。このため、システム全体の性能を向上させることが可能となる。

【０１４５】本発明の第３の態様の構成によれば、ファイルの最終ブロックにアクセスするのでなければ、サイズトークンを獲得することなくファイルにアクセスすることが可能となり、これと並行して、他のノードは、サイズトークンを獲得してファイルの最終ブロックにアクセスし、ファイルのサイズを拡張するwrite 操作処理を実行することができる。このため、例えばファイルを拡張するプログラムとファイルをその先頭から順に読むプログラムをそれぞれ異なるノードで同時に実行させることが可能となり、システム全体の性能を向上させることができる。

【０１４６】本発明の第４の態様の構成によれば、複数のノードは、ディスク装置内のファイルに、ＬＡＮ経由ではなく直結された制御・データ線を介してアクセスすることが可能となる。

【０１４７】本発明の第５の態様の構成によれば、サーバ切替時にも、矛盾のないファイル時刻の付与が可能となる。本発明の第６の態様の構成によれば、クライアント装置は、サーバ装置に問い合わせることなく、新たなブロックをファイルに割り当てることが可能となる。このため、クライアント装置とサーバ装置との間の通信回数を削減でき、システム全体の性能を向上させることが可能となる。更に、サイズ毎にリザーブすることにより最適な連続ブロックを割り当てフラグメンテーションを防止すると共にファイルアクセス性能を向上させることができる。また、新たに割り当てられたエクステントは、データが書き込まれた後のクライアント装置からサーバ装置への応答によって初めて、そのファイルのメタデータ等として記憶される。このため、悪意をもってデータを覗くことを防止することが可能となる。

【０１４８】本発明の第７の構成によれば、トークン制御において、デッドロックの発生を回避することのできる効率的なファイルロック制御が実現される。本発明の第８の構成によれば、トークンに基づいてトランザクション制御されているメタデータ等の更新処理におけるデッドロックの発生を適切に検出することができる。

【０１４９】本発明の第９の構成によれば、トランザクションのキャンセルに伴う常駐制御表の高速なリストアが実現される。本発明の第１０の構成によれば、ログファイルに書き出されるログデータ量の削減が実現される。

【０１５０】本発明の第１１の構成によれば、ログファイルを実ディスク上に書き出すログフラッシュ処理を、実行中のトランザクションと独立して行うことが可能となり、システム性能の向上が実現される。

【図面の簡単な説明】

【図１】本発明の実施の形態のシステム構成図である。

【図２】ノード内のソフトウェア構成図である。

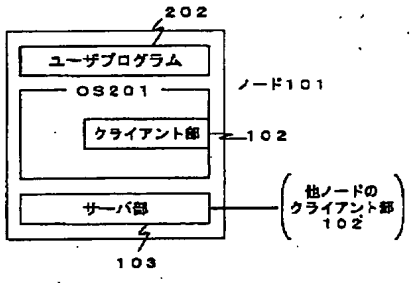
【図３】クライアント部のメイン動作フローチャートで

ある。
【図４】クライアント部のopen操作処理の動作フローチャートである。
【図５】サーバ部のメイン動作フローチャート（その１）である。
【図６】サーバ部のメイン動作フローチャート（その２）である。
【図７】サーバ部のopen操作処理の動作フローチャートである。
【図８】クライアント部のread/write操作処理の動作フローチャートである。
【図９】クライアント部のファイル時刻操作処理の動作フローチャートである。
【図１０】サーバ部でのread権の時刻トークンの応答処理の動作フローチャートである。
【図１１】サーバ部でのwrite 権の時刻トークンの応答処理の動作フローチャートである。
【図１２】サーバ部でのデータトークンの応答処理の動作フローチャートである。
【図１３】エクステント管理の詳細を示す図である。
【図１４】エクステント管理のシーケンス図である。
【図１５】ログ制御機構を実装したノード間ファイル共有管理システムの基本構成図である。
【図１６】ロック継承制御処理の説明図である。
【図１７】デッドロック検出処理の説明図である。
【図１８】ログファイルの２次キャッシュ制御の説明図である。
【図１９】ログデータ量を削減できるログ制御処理の説明図である。
【図２０】トランザクションキャンセル時におけるメモリ常駐制御表のリストア処理の説明図である。

【符号の説明】
１０１、１５０３　　ノード
１０２　　クライアント部
１０３　　サーバ部
１０４、１５０２　　メタデータ
１０５　　ファイル
１０６　　ＬＡＮ
２０１　　オペレーティングシステム（ＯＳ）
２０２　　ユーザプログラム
１５０１　　共用ファイル管理装置
１５０４　　バッファキャッシュ
１５０５　　ログファイル
１６０１　　トークン回収待ちキュー
１６０２　　トークン回収制御表
１７０１、２００２　　ファイル制御表
１７０１ａ、１７０２ａ　　ファイルロック
１７０１ｂ、１７０２ｂ　　オーナ
１７０２　　バッファキャッシュ制御表
１７０３　　スレッド制御表
１７０３ａ　　待ちリソース
１７０３ｂ　　タイプ
１８０１　　２次キャッシュ
１９０１　　ログキュー
１９０２　　ログ制御表
２００１　　ファイルロックキュー
２００３　　常駐制御表

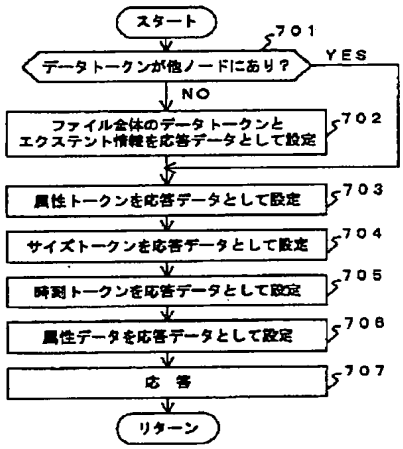
【図２】

ノード内のソフトウェア構成図



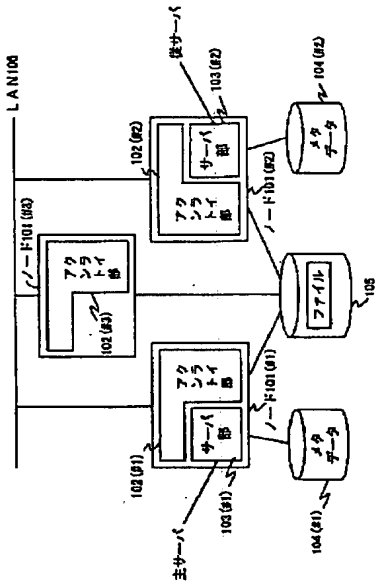
【図７】

サーバ部のopen操作処理の動作フローチャート



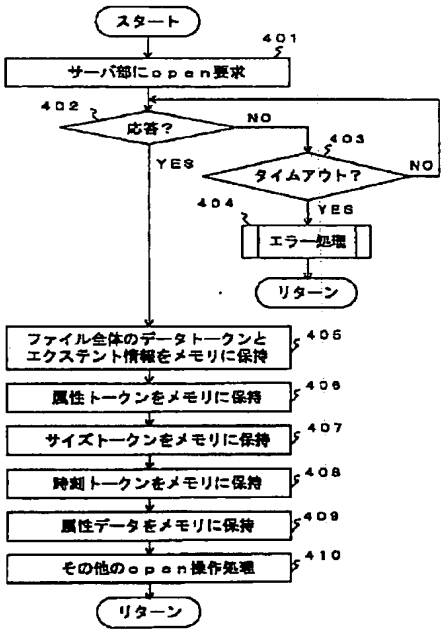
【図１】

本発明の実施の形態のシステム構成図



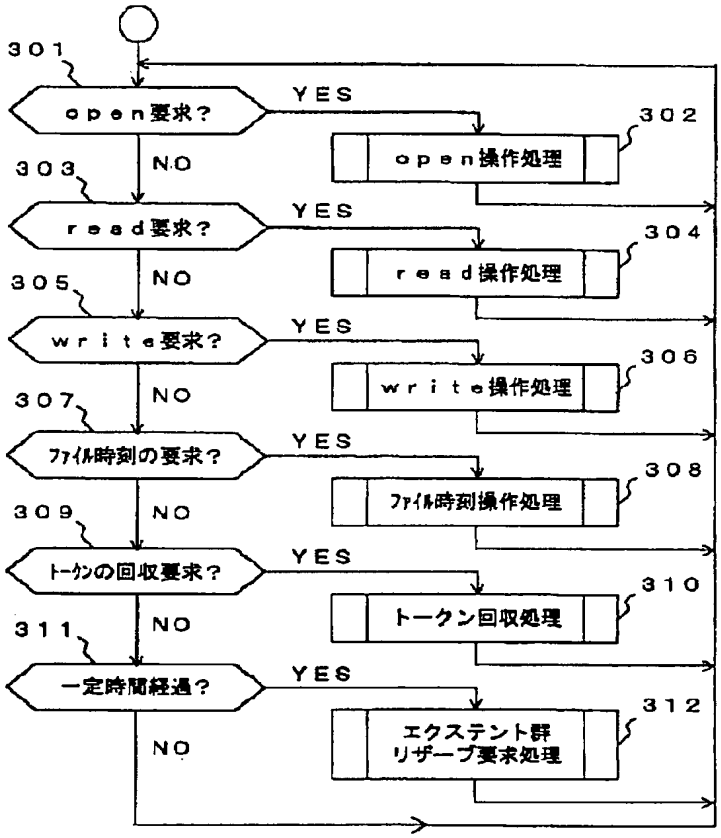
【図４】

クライアント部のｏｐｅｎ操作処理の動作フローチャート



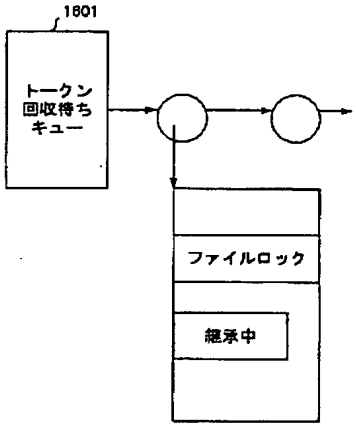
【図３】

クライアント部のメイン動作フローチャート



【図１６】

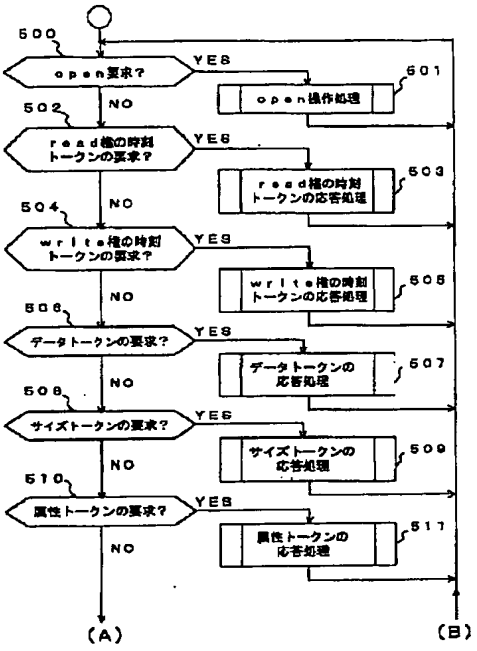
ロック継承制御処理の説明図



トークン回収制御表 1602

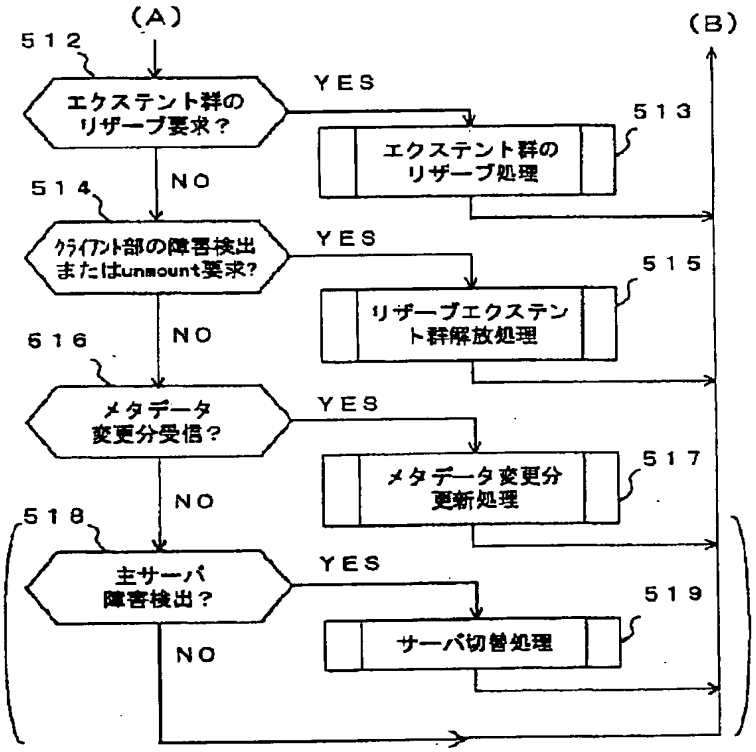
【図５】

サーバ部のメイン動作フローチャート（その１）



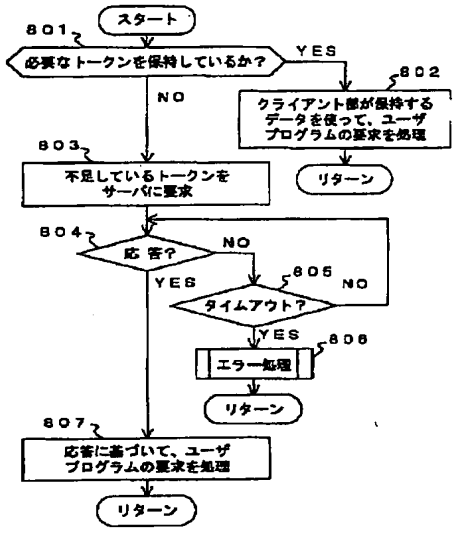
【図６】

サーバ部のメイン動作フローチャート（その２）



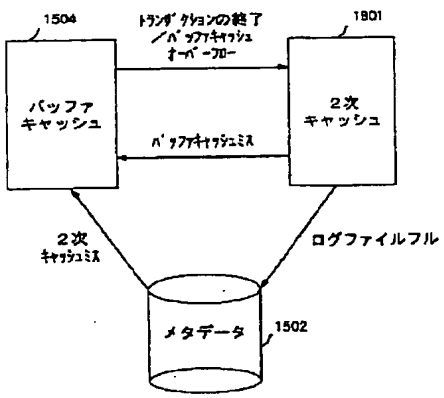
【図８】

クライアント部のread/write操作処理の動作フローチャート



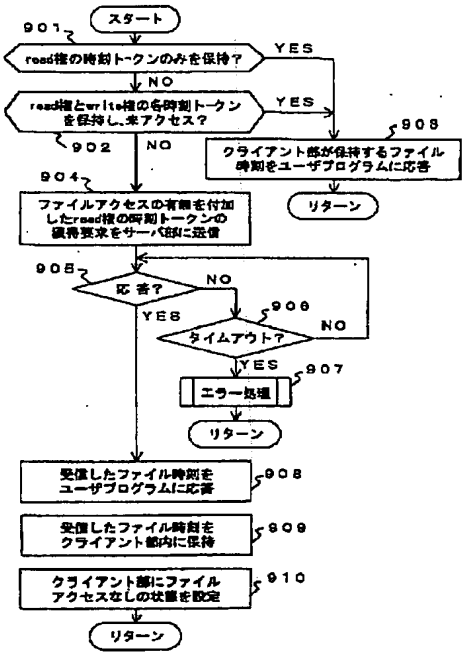
【図１８】

ログファイルの２次キャッシュ制御の説明図



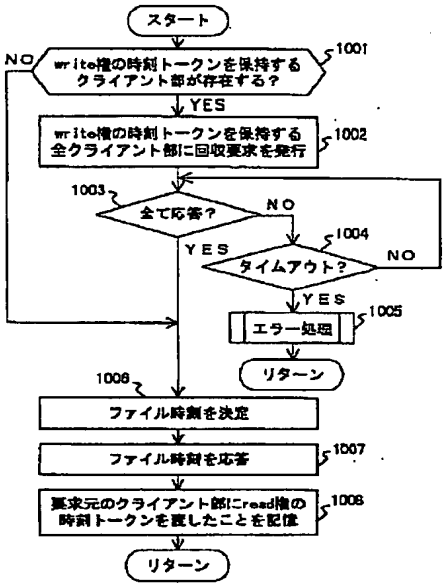
【図９】

クライアント部のファイル時刻操作処理の
動作フローチャート



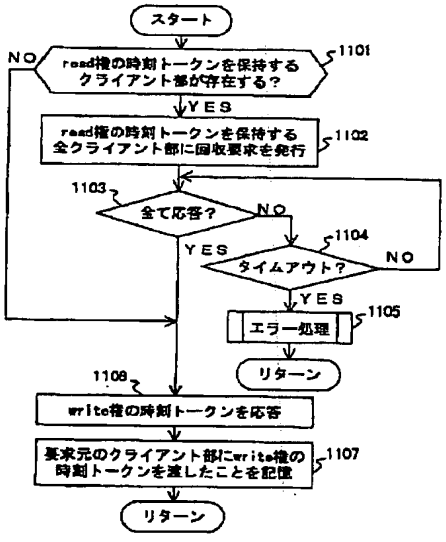
【図１０】

サーバ部でのread権の時刻トークンの
応答処理の動作フローチャート



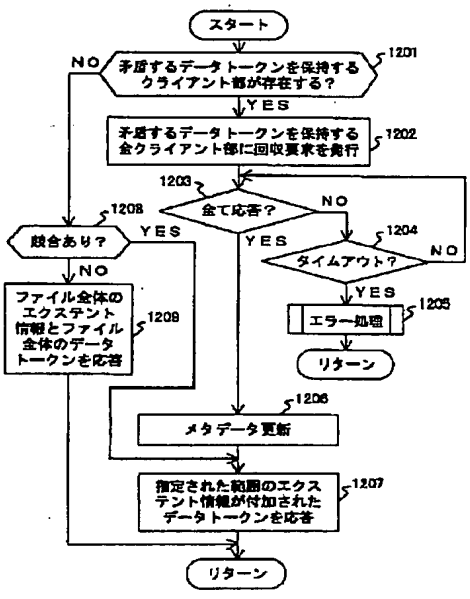
【図１１】

サーバ部でのwrite権の時刻トークンの
応答処理の動作フローチャート



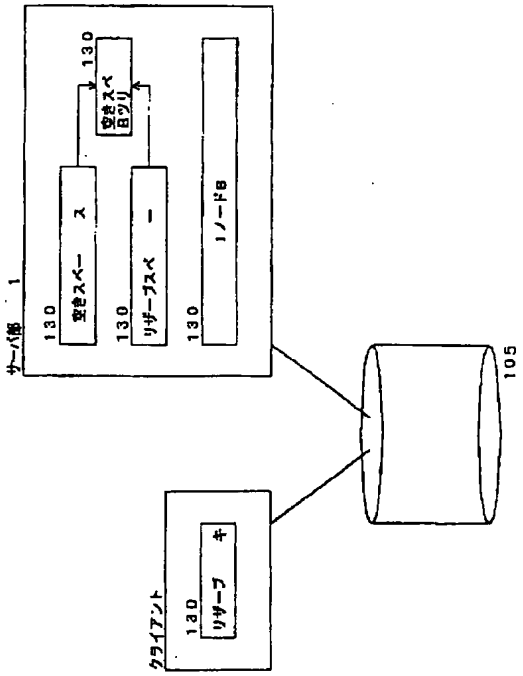
【図１２】

サーバ部でのデータトークンの
応答処理の動作フローチャート



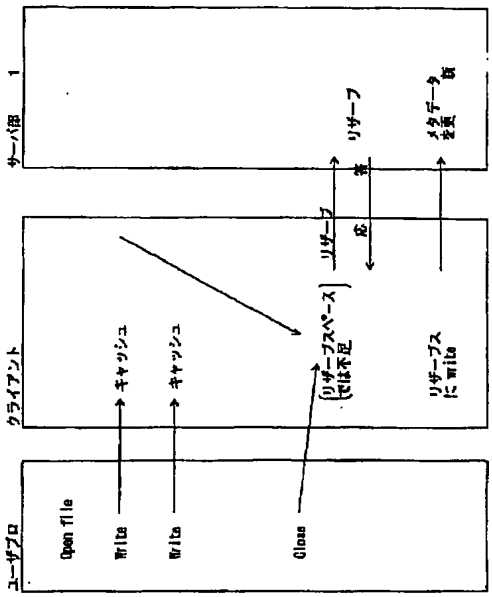
【図１３】

エクステント管理の詳細を示す図



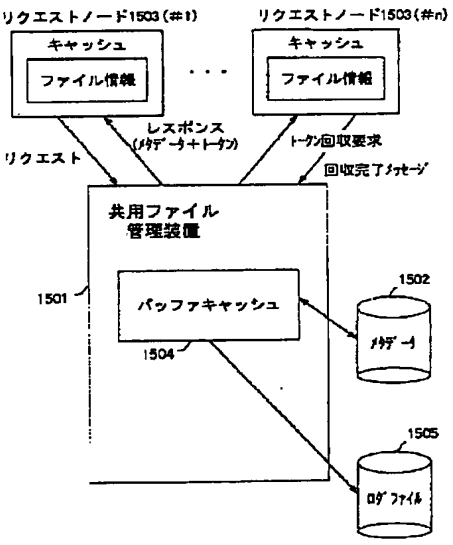
【図１４】

エクステント管理のシーケンス図



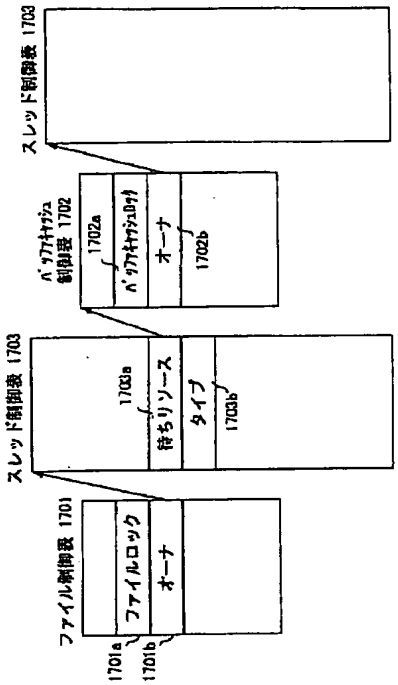
【図１５】

ログ制御機構を実装した
ノード間ファイル共用管理システムの基本構成図



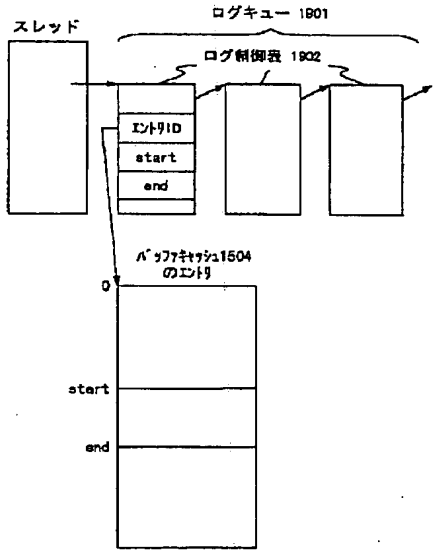
【図１７】

デッドロック検出処理の説明図



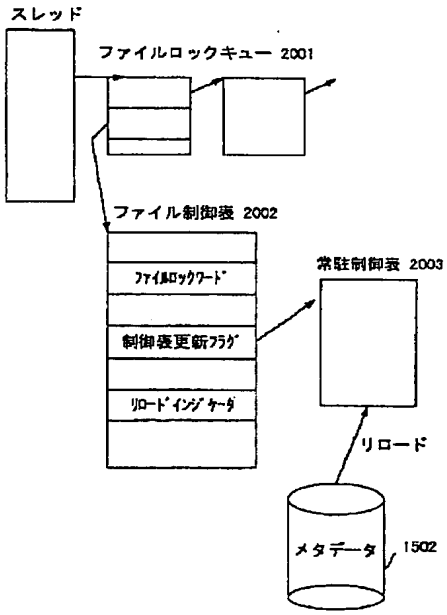
【図１９】

ログデータ量を削減できるログ制御処理の説明図



【図２０】

トランザクションキャンセル時における
メモリ常駐制御表のリトスア制御処理の説明図



フロントページの続き

(72)発明者　村上　岳生
神奈川県川崎市中原区上小田中４丁目１番
１号　富士通株式会社内

Ｆターム(参考)　5B082 DE03 EA07 FA18 HA02
5B089 GA11 GA21 GB02 HA06 JB15
KA06 KB09 KB11